PEARSON

Multimedia Communications

Applications, Networks, Protocols and Standards

Fred Halsall

This edition is manufactured in India and is authorized for sale only in India, Bangladesh. Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is UNAUTHORIZED.

right of Fred Halsall to be identified as author of this Work has been asserted by him in accordance
ı the Copyright, Designs and Patents Act 1988.

yright © 2001 by Pearson Education, Ltd.
s edition is published by arrangement with Pearson Education, Ltd.

; book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold,
d out, or otherwise circulated without the publisher's prior written consent in any form of binding or
er other than that in which it is published and without a similar condition including this condition being
osed on the subsequent purchaser and without limiting the rights under copyright reserved above, no
of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in
form or by any means (electronic, mechanical, photocopying, recording or otherwise), without the prior
ten permission of both the copyright owner and the above-mentioned publisher of this book.

N 978-81-317-0994-8

t Impression, 2007
ond Impression, 2008
d Impression, 2008
rth Impression, 2009

***s edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan,
istan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is
UTHORIZED.***

lished by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia.

d Office: 482, F.I.E., Patparganj, Delhi 110 092, India.
istered Office: 14 Local Shopping Centre, Panchsheel Park, New Delhi 110 017, India.

ted in India by Saurbh Printers.

short contents

Preface
Acknowledgments

chapter 1 **multimedia communications** 23

1.1 Introduction 23
1.2 Multimedia information representation 24
1.3 Multimedia networks 25
1.4 Multimedia applications 34
1.5 Application and networking terminology 53
1.6 Summary 71

chapter 2 **multimedia information representation** 79

2.1 Introduction 79
2.2 Digitization principles 80
2.3 Text 89
2.4 Images 96
2.5 Audio 110
2.6 Video 118
2.7 Summary 134

chapter 3 **text and image compression** 138

3.1 Introduction 138
3.2 Compression principles 139
3.3 Text compression 146
3.4 Image compression 162
3.5 Summary 190

chapter 4 **audio and video compression** 195

4.1 Introduction 195
4.2 Audio compression 195
4.3 Video compression 215
4.4 Summary 255

chapter 5 **standards for multimedia communications** 261

5.1 Introduction 261
5.2 Reference models 262
5.3 Standards relating to interpersonal communications 268
5.4 Standards relating to interactive applications over the Internet 290
5.5 Standards for entertainment applications 275
5.6 Summary 283

chapter 6 **digital communication basics** 309

6.1 Introduction 309
6.2 Transmission media 313
6.3 Sources of signal impairment 324
6.4 Asynchronous transmission 333
6.5 Synchronous transmission 340
6.6 Error detection methods 349
6.7 Protocol basics 358
6.8 The HDLC protocol 385
6.9 Summary 393

chapter 7 **circuit-switched networks** 400

7.1 Introduction 400
7.2 Transmission systems 404
7.3 Switching systems 440
7.4 Signaling systems 448
7.5 Summary 479

chapter 8 **enterprise networks** 474

8.1 Introduction 474
8.2 LANs 475
8.3 Ethernet/IEEE802.3 476
8.4 Token ring 485

; 500
516
peed LANs 531
rotocols 543
te LAN interconnection
logies 549
ary 559

the internet 564
uction 564
agrams 568
ientation and reassembly 571
lresses 573
nd RARP 578
ng algorithms 583
625
upport 629
PP link layer protocol 637
639
IPv4 interoperability 655
nary 660

0 **broadband ATM networks** 668
duction 668
format and switching principles 670
h architectures 673
ocol architecture 679
[LANs 689
[MANs 703
e area ATM networks 724
mary 727

11 **entertainment networks and high-speed modems** 731
oduction 731
le TV networks 732
ellite television networks 759
restrial television networks 771
h-speed PSTN access technologies 778
nmary 785

chapter 12 **transport protocols** 791
12.1 Introduction 791
12.2 TCP/IP protocol suite 792
12.3 TCP 794
12.4 UDP 836
12.5 RTP and RTCP 842
12.6 Summary 848

chapter 13 **application support functions** 852
13.1 Introduction 852
13.2 ASN.1 854
13.3 Security 867
13.4 Data encryption 867
13.5 Nonrepudiation 883
13.6 Authentication 885
13.7 Public key certification authorities 890
13.8 Summary 891

chapter 14 **internet applications** 896
14.1 Introduction 896
14.2 Domain name system 897
14.3 Electronic mail 908
14.4 FTP 925
14.5 TFTP 931
14.6 Internet telephony 935
14.7 SNMP 941
14.8 Summary 949

chapter 15 **the World Wide Web** 955
15.1 Introduction 955
15.2 URLs and HTTP 956
15.3 HTML 966
15.4 Audio and video 987
15.5 Java and JavaScript 993
15.6 Security 1000
15.7 Web operation 1006
15.8 Summary 1009
Appendix A CRC implementation 1015
Appendix B Forward error control 1019
Bibliography 1030
Index 1041

contents

chapter 1 **multimedia communications** 23
1.1 Introduction 23
1.2 Multimedia information representation 24
1.3 Multimedia networks 25
  1.3.1 Telephone networks 26
  1.3.2 Data networks 28
  1.3.3 Broadcast television networks 30
  1.3.4 Integrated services digital networks 32
  1.3.5 Broadband multiservice networks 33
1.4 Multimedia applications 34
  1.4.1 Interpersonal communications 35
  1.4.2 Interactive applications over the Internet 46
  1.4.3 Entertainment applications 49
1.5 Application and networking terminology 53
  1.5.1 Media types 53
  1.5.2 Communication modes 54
  1.5.3 Network types 56
  1.5.4 Multipoint conferencing 61
  1.5.5 Network QoS 63
  1.5.6 Application QoS 67
1.6 Summary 71
Exercises 75

chapter 2 **multimedia information representation** 79
2.1 Introduction 79
2.2 Digitization principles 80
  2.2.1 Analog signals 80
  2.2.2 Encoder design 82
  2.2.3 Decoder design 88
2.3 Text 89
  2.3.1 Unformatted text 89
  2.3.2 Formatted text 91
  2.3.3 Hypertext 93
2.4 Images 96
  2.4.1 Graphics 96
  2.4.2 Digitized documents 99

- 2.4.3 Digitized pictures 100
- 2.5 Audio 110
  - 2.5.1 PCM speech 112
  - 2.5.2 CD-quality audio 115
  - 2.5.3 Synthesized audio 115
- 2.6 Video 118
  - 2.6.1 Broadcast television 119
  - 2.6.2 Digital video 114
  - 2.6.3 PC video 132
  - 2.6.4 Video content 133
- 2.7 Summary 134
- Exercises 135

chapter 3 text and image compression 138

- 3.1 Introduction 138
- 3.2 Compression principles 139
  - 3.2.1 Source encoders and destination decoders 139
  - 3.2.2 Lossless and lossy compression 139
  - 3.2.3 Entropy encoding 140
  - 3.2.4 Source encoding 143
- 3.3 Text compression 146
  - 3.3.1 Static Huffman coding 146
  - 3.3.2 Dynamic Huffman coding 152
  - 3.3.3 Arithmetic coding 156
  - 3.3.4 Lempel–Ziv coding 158
  - 3.3.5 Lempel–Ziv-Welsh coding 159
- 3.4 Image compression 162
  - 3.4.1 Graphics interchange format 163
  - 3.4.2 Tagged image file format 164
  - 3.4.3 Digitized documents 166
  - 3.4.4 Digitized pictures 172
  - 3.4.5 JPEG 172
- 3.5 Summary 190
- Exercises 192

chapter 4 audio and video compression 195

- 4.1 Introduction 195
- 4.2 Audio compression 195
  - 4.2.1 Differential pulse code modulation 196
  - 4.2.2 Adaptive differential PCM 199
  - 4.2.3 Adaptive predictive coding 201
  - 4.2.4 Linear predictive coding 201
  - 4.2.5 Code-excited LPC 203
  - 4.2.6 Perceptual coding 204
  - 4.2.7 MPEG audio coders 207
  - 4.2.8 Dolby audio coders 212
- 4.3 Video compression 215
  - 4.3.1 Video compression principles 216
  - 4.3.2 H.261 225
  - 4.3.3 H.263 229
  - 4.3.4 MPEG 237
  - 4.3.5 MPEG-1 238
  - 4.3.6 MPEG-2 242
  - 4.3.7 MPEG-4 246
- 4.4 Summary 255
- Exercises 257

chapter 5 standards for multimedia communications 261

- 5.1 Introduction 261
- 5.2 Reference models 262
  - 5.2.1 TCP/IP reference model 265
  - 5.2.2 Protocol basics 267
- 5.3 Standards relating to interpersonal communications 268
  - 5.3.1 Circuit-mode networks 270
  - 5.3.2 Packet-switched networks 278
  - 5.3.3 Electronic mail 286
- 5.4 Standards relating to interactive applications over the Internet 290
  - 5.4.1 Information browsing 291
  - 5.4.2 Electronic commerce 293
  - 5.4.3 Intermediate systems 295
  - 5.4.4 Java and JavaScript 296
- 5.5 Standards for entertainment applications 297
  - 5.5.1 Movie/video-on-demand 298
  - 5.5.2 Interactive television 303
- 5.6 Summary 305
- Exercises 307

chapter 6 digital communication basics 309

- 6.1 Introduction 309
- 6.2 Transmission media 313
  - 6.2.1 Two-wire open lines 314
  - 6.2.2 Twisted-pair lines 314

6.2.3 Coaxial cable 315
6.2.4 Optical fiber 316
6.2.5 Satellites 318
6.2.6 Terrestrial microwave 320
6.2.7 Radio 320
6.2.8 Signal propagation delay 321
6.3 Sources of signal impairment 324
6.3.1 Attenuation 325
6.3.2 Limited bandwidth 326
6.3.3 Delay distortion 330
6.3.4 Noise 331
6.4 Asynchronous transmission 333
6.4.1 Bit synchronization 336
6.4.2 Character synchronization 336
6.4.3 Frame synchronization 338
6.5 Synchronous transmission 340
6.5.1 Bit synchronization 340
6.5.2 Character-oriented 346
6.5.3 Bit-oriented 348
6.6 Error detection methods 349
6.6.1 Parity 350
6.6.2 Block sum check 352
6.6.3 Cyclic redundancy check 354
6.7 Protocol basics 358
6.7.1 Error control 359
6.7.2 Idle RQ 359
6.7.3 Continous RQ 364
6.7.4 Flow control 371
6.7.5 Sequence numbers 371
6.7.6 Layered architecture 373
6.7.7 Protocol specification 376
6.7.8 User service primitives 381
6.8 The HDLC protocol 385
6.9 Summary 393
Exercises 396

chapter 7 **circuit-switched networks** 400

7.1 Introduction 404
7.2 Transmission systems 404
7.2.1 Analog subscriber lines 404
7.2.2 PSTN modems 408
7.2.3 Digital subscriber lines 420
7.2.4 Plesiochronous digital hierarchy 430

7.2.5 Synchronous digital hierarchy 434
7.3 Switching systems 440
7.3.1 Time switching 441
7.3.2 Space switching 443
7.3.3 Digital switching units 446
7.4 Signaling systems 448
7.4.1 Access network signaling 449
7.4.2 Trunk network signaling 461
7.5 Summary 468
Exercises 470

chapter 8 **enterprise networks** 474

8.1 Introduction 474
8.2 LANs 475
8.3 Ethernet/IEEE802.3 476
8.4 Token ring 485
8.5 Bridges 500
8.5.1 Transparent bridges 501
8.5.2 Source routing bridges 511
8.6 FDDI 516
8.7 High-speed LANs 531
8.7.1 Fast Ethernet 531
8.7.2 Switched Fast Ethernet 539
8.7.3 Gigabit Ethernet 542
8.8. LAN protocols 543
8.8.1 Physical layer 544
8.8.2 MAC sublayer 545
8.8.3 LLC sublayer 545
8.8.4 Network layer 549
8.9 Multisite LAN interconnection technologies 549
8.9.1 Intersite gateways 550
8.9.2 ISDN switched connections 551
8.9.3 Frame relay 554
8.9.4 High bit rate leased lines 557
8.10 Summary 559
Exercises 560

chapter 9 **the Internet** 564

9.1 Introduction 564
9.2 IP datagrams 568
9.3 Fragmentation and reassembly 571

9.4 IP addresses 573
  9.4.1 Subnets 576
9.5. ARP and RARP 578
  9.5.1 ARP 578
  9.5.2 RARP 580
  9.5.3 ARP/RARP message formats and transmission 581
9.6 Routing algorithms 583
  9.6.1 Static routing 586
  9.6.2 Flooding 588
  9.6.3 Distance vector routing 589
  9.6.4 Link-state shortest-path-first routing 592
  9.6.5 Hierarchical routing 599
  9.6.6 Classless inter-domain routing 605
  9.6.7 Tunneling 608
  9.6.8 Broadcast routing 609
  9.6.9 Multicast routing 615
  9.6.10 IGMP 622
  9.6.11 M-bone 624
9.7 ICMP 625
  9.7.1 ICMP message formats and transmission 629
9.8 QoS support 629
  9.8.1 Integrated services 630
  9.8.2 Differentiated services 635
9.9 The PPP link layer protocol 637
9.10 IPv6 639
  9.10.1 Datagram format 640
  9.10.2 Address structure 643
  9.10.3 Extension headers 648
  9.10.4 Autoconfiguration 654
9.11 IPv6/IPv4 interoperability 655
  9.11.1 Dual protocols 656
  9.11.2 Dual stacks and tunneling 656
  9.11.3 Translators 658
9.12 Summary 660
  Exercises 662

chapter 10 broadband ATM networks 668

10.1 Introduction 668
10.2 Cell format and switching principles 670
10.3 Switch architectures 673
10.4 Protocol architecture 679
  10.4.1 ATM adaptation layer 680
  10.4.2 ATM layer 687

10.5 ATM LANs 689
  10.5.1 Call processing 695
10.6 ATM MANs 703
  10.6.1 Subnetwork architectures 705
  10.6.2 Protocol architecture 707
  10.6.3 Queued arbitrated access protocol 710
  10.6.4 Bandwidth balancing 712
  10.6.5 Prioritized distributed queuing 715
  10.6.6 Slot and segment formats 718
  10.6.7 SMDS 720
10.7 Wide area ATM networks 724
10.8 Summary 727
  Exercises 728

chapter 11 entertainment networks and high-speed modems 731

11.1 Introduction 731
11.2 Cable TV networks 732
  11.2.1 HFC networks 736
11.3 Satellite television networks 759
  11.3.1 Broadcast television principles 761
  11.3.2 Digital television 766
  11.3.3 Interactive services 770
11.4 Terrestrial television networks 771
  11.4.1 Broadcast television principles 771
  11.4.2 Digital television 773
  11.4.3 Interactive services 778
11.5 High-speed PSTN access technologies 778
  11.5.1 ADSL 779
  11.5.2 VDSL 784
11.6 Summary 785
  Exercises 787

chapter 12 transport protocols 791

12.1 Introduction 791
12.2 TCP/IP protocol suite 792
12.3 TCP 794
  12.3.1 User services 795
  12.3.2 Protocol operation 800
  12.3.3 Additional features 823
  12.3.4 Protocol specification 833

12.4 UDP 836
  12.4.1 User services 838
  12.4.2 Protocol operation 840
12.5 RTP and RTCP 842
  12.5.1 RTP 842
  12.5.2 RTCP 845
12.6 Summary 847
  Exercises 848

chapter 13 application support functions 852

13.1 Introduction 852
13.2 ASN.1 854
  13.2.1 Type definitions 856
  13.2.2 Transfer syntax 861
13.3 Security 867
13.4 Data encryption 867
  13.4.1 Terminology 868
  13.4.2 Basic techniques 869
  13.4.3 The data encryption standard 872
  13.4.4 IDEA 879
  13.4.5 The RSA algorithm 881
13.5 Nonrepudiation 883
13.6 Authentication 885
  13.6.1 Using a public key system 885
  13.6.2 Using a private key system 886
13.7 Public key certification authorities 890
13.8 Summary 891
  Exercises 893

chapter 14 internet applications 896

14.1 Introduction 896
14.2 Domain name system 897
  14.2.1 Name structure and administration 897
  14.2.2 DNS resource records 900
  14.2.3 DNS query messages 902
  14.2.4 Name servers 902
  14.2.5 Service requests 903
14.3 Electronic mail 908
  14.3.1 Structure of email messages 909
  14.3.2 Message content 912
  14.3.3 Message transfer 921
14.4 FTP 925
  14.4.1 Overview 925
  14.4.2 File content representation 925
  14.4.3 FTP operation 926
  14.4.4 Command and reply message format 927
  14.4.5 Example 929
  14.4.6 Anonymous FTP 931
14.5 TFTP 931
  14.5.1 Protocol 932
14.6 Internet telephony 935
  14.6.1 SIP 937
  14.6.2 SDP 939
  14.6.3 GLP 939
14.7 SNMP 941
  14.7.1 Structure of management information 943
  14.7.2 Protocol 947
14.8 Summary 949
  Exercises 951

chapter 15 The World Wide Web 955

15.1 Introduction 955
15.2 URLs and HTTP 956
  15.2.1 URLs 957
  15.2.2 HTTP 960
15.3 HTML 966
  15.3.1 Text format directives 967
  15.3.2 Lists 970
  15.3.3 Color 972
  15.3.4 Images and lines 973
  15.3.5 Tables 977
  15.3.6 Forms and CGI scripts 978
  15.3.7 Web mail 983
  15.3.8 Frames 983
15.4 Audio and video 987
  15.4.1 Streaming using a Web server 988
  15.4.2 Streaming servers and RTSP 990
15.5 Java and JavaScript 993
  15.5.1 Java 995
  15.5.2 JavaScript 997
15.6 Security 1000
  15.6.1 SSL 1000
  15.6.2 SET 1004
15.7 Web operation 1006

15.7.1 Search Engines 1006
15.7.2 Portals 1008
15.8 Summary 1009
Exercises 1011

endix A CRC implementation 1015
endix B Forward error control 1019
ography 1030
Index 1041

preface

Objectives

Multimedia communications embraces a range of applications and networking infrastuctures. The term multimedia is used to indicate that the information/data relating to an application may be composed of a number of different types of media which are integrated together in some way. The different media types are text, images, speech, audio and video and some example applications are video telephony (speech and video), multimedia electronic mail (text, images and audio for example), interactive television (text, audio and video), electronic commerce (text, images, audio and video), Web TV (text, audio and video) and many others.

In practice, there are a number of different types of network that are used to provide the networking infrastructure. These include not only networks that were designed from the outset to provide multimedia communication services – normally referred to as broadband multiservice networks – but also networks that were designed initially to provide just a single type of service and it is as a result of advances in various technologies that these can now support a range of other (multimedia) services. For example, public (and private) switched telephone networks (PSTNs) were designed initially to provide a basic telephony service but they are now used to support a range of more advanced multimedia applications involving all of the different media types. Similarly, computer networks such as the Internet, which were designed initially to provide general data communication services such as electronic mail and file transfers, can now support a much richer set of multimedia applications.

In terms of the different types of media, text and images are generated and represented in a digital form. Speech, audio and video, however, are generated in the form of continuously varying – normally referred to as analog - signals. Hence in order to integrate all of the different media types together, it is necessary to first convert the various analog signals into a digital form. The integrated digital information stream can then be stored within a computer and transmitted over a network in a unified way. In addition, unlike text and images which are created in the form of a single block of digital information, since speech, audio and video are continuously varying signals, the digitization process can produce large volumes of information which carries on increasing with time. Hence in most multimedia applications, in

The applications may involve either person-to-person communications or person-to-system communications. In general, two people communicate with each other through suitable **terminal equipment (TE)** while a person interacts with a system using either a **multimedia personal computer (PC)** or **workstation.** Typically, these are located either in the home or on a desktop in an office and the system is a **server** containing a collection of files or documents each comprising digitized text, images, audio, and video information either singly or integrated together in some way. Alternatively, the server may contain a library of digitized movies/videos and the user interacts with the server by means of a suitable selection device that is connected to the **set-top box (STB)** associated with a television.

In practice, there are a number of different types of network that are used to provide the networking infrastructure. These include not only networks that were designed from the outset to provide multimedia communication services but also networks that were designed initially to provide just a single type of service and it is as a result of advances in various technologies that these can now provide a range of other services. For example, **public switched telephone networks (PSTNs)** – also known as **general switched telephone networks (GSTNs)** – were designed initially to provide a basic switched telephone service but, as a result of advances in digital signal processing hardware and associated software, they now provide a range of more advanced services involving text, images, and video. Similarly, data networks that were designed initially to support basic data applications such as electronic mail and file transfers, now support a much richer set of applications that involve images, audio, and video.

In this chapter we shall present an overview of, firstly, how the different media types are represented, secondly, the different types of network that are used to provide multimedia communication services, and thirdly, a selection of the applications that these networks support. Finally, we describe the meaning of a range of terms that are associated with multimedia communications.

1.2 Multimedia information representation

Applications involving text and images comprise **blocks** of digital data. In the case of text, for example, a typical unit is a block of characters with each character represented by a fixed number of **binary digits (bits)** known as a **codeword**. Similarly, a digitized image comprises a two-dimensional block of what are called **picture elements** with each element represented by a fixed number of bits. Also, since a typical application involving text and images comprises a short request for some information – a file, for example – and the file contents being returned, the duration of the overall transaction is relatively short.

In applications involving audio and video, however, the audio and video signals vary continuously with time as the amplitude of the speech, audio, or video signal varies. This type of signal is known as an **analog signal** and, typically, the duration of applications that involve audio and/or video can be relatively long. A typical telephone conversation, for example, can last for several minutes while a movie (comprising audio and video) can last for a number of hours.

In applications that involve just a single type of media, the basic form of representation of the particular media type is often used. Similarly, in applications that involve either text-and-images or audio-and-video their basic form is often used since the two media types in these applications have the same form of representation. However, in applications that involve the different media types integrated together in some way, it becomes necessary to represent all four media types in a digital form. In the case of text and images, this is their standard form of representation. For audio and video, however, because their basic forms of representation are analog signals, these must be converted into a corresponding digital form before they can be integrated with the two other media types.

As we shall describe in the next chapter, the digitization of an audio signal produces a digital signal which, because the amplitude of the signal varies continuously with time, is of a relatively high bit rate. This is measured in **bits per second (bps)** and, in the case of a speech signal, for example, a typical bit rate is 64 kbps. Moreover, because applications involving audio can be of a long duration, this bit rate must be sustained over an equally long time period. The same applies to the digitization of a video signal except that much higher bit rates and longer time durations are involved. In general, however, as we shall expand upon in the next section, the communication networks that are used to support applications that involve audio and video cannot support the very high bit rates that are required for representing these media types in a digital form. As a result, a technique known as **compression** is first applied to the digitized signals in order to reduce the resulting bit rate to a level which the various networks can support. Compression is also applied to text and images in order to reduce the time delay between a request being made for some information and the information becoming available on, say, the screen of a computer. We shall describe a selection of the compression algorithms that are used with text and images in Chapter 3 and those used with audio and video in Chapter 4.

1.3 Multimedia networks

There are five basic types of communication network that are used to provide multimedia communication services:

- telephone networks,
- data networks,
- broadcast television networks,
- integrated services digital networks,
- broadband multiservice networks.

As the names imply, the first three network types were initially designed to provide just a single type of service: telephony, data communications, and broadcast television respectively. The last two network types, however, were designed from the outset to provide multiple services. We shall describe the essential features of each type of network separately and, in the case of the first three network types, the technological developments that have enabled them to provide additional services.

1.3.1 Telephone networks

Public switched telephone networks have been in existence for many years and have gone through many changes during this time. They were designed to provide a basic switched telephone service which, with the advent of the other network types, has become known as a **plain old telephone service** or **POTS**. The term "switched" is used to indicate that a subscriber can make a call to any other telephone that is connected to the total network. Initially, such networks spanned just a single country but later the telephone networks of different countries were interconnected so that they now provide an international switched service. The main components of the network are shown in diagrammatic form in Figure 1.1(a).

As we can see, telephones located in the home or in a small business are connected directly to their nearest **local exchange/end office.** Those located in a medium or large office/site are connected to a private switching office known as a **private branch exchange** or **PBX**. The PBX provides a (free) switched service between any two telephones that are connected to it. In addition, the PBX is connected to its nearest local (public) exchange which enables the telephones that are connected to the PBX also to make calls through a PSTN. More recently, **cellular phone networks** have been introduced which provide a similar service to mobile subscribers by means of handsets that are linked to the cellular phone network infrastructure by radio. The switches used in a cellular phone network are known as **mobile switching centers (MSCs)** and these, like a PBX, are also connected to a switching office in a PSTN which enables both sets of subscribers to make calls to one another. Finally, international calls are routed to and switched by **international gateway exchanges (IGEs)**.

As we indicated earlier, a speech signal is an analog signal since it varies continuously with time according to the amplitude and frequency variations of the sound resulting from the speech. A microphone is used to convert this into an analog electrical signal. Because of this, telephone networks operate in what is called a **circuit mode** which means that, for each call, a separate circuit is set up through the network – of the necessary capacity – for the duration of the call. The **access circuits** that link the telephone handsets to a PSTN or PBX were designed, therefore, to carry the two-way analog signals associated with a call. Hence, although within a PSTN all the switches and the transmission circuits that interconnect them now operate in a **digital mode**,

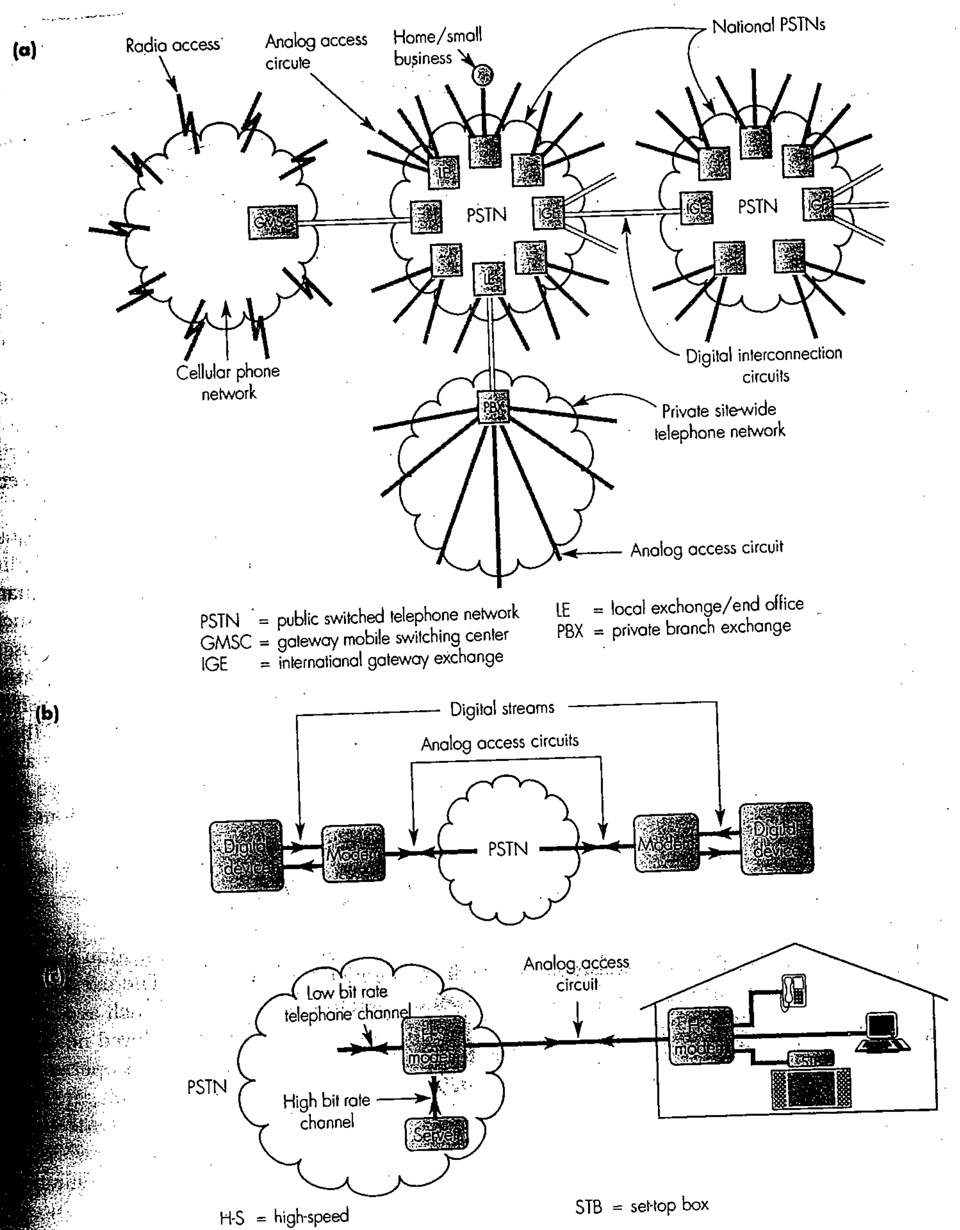

(a)
Radio access
Analog access circuite
Home/small business
National PSTNs
GMSC
LE
PSTN
IGE
PBX
Cellular phone network
Digital interconnection circuits
Private site-wide telephone network
Analog access circuit
PSTN = public switched telephone network
GMSC = gateway mobile switching center
IGE = international gateway exchange
LE = local exchange/end office
PBX = private branch exchange
(b)
Digital streams
Analog access circuits
Digital device
Modem
PSTN
Modem
Digital device
(c)
Low bit rate telephone channel
H-S modem
High bit rate channel
Server
PSTN
Analog access circuit
H-S modem
STB
H-S = high-speed
STB = set-top box

Figure 1.1 Telephone networks: (a) network components; (b) digital transmission using modems; (c) multiple services via an H-S modem.

to carry a digital signal – a stream of binary 1s and 0s – over the analog access circuits requires a device known as a **modem**. The general scheme is shown in Figure 1.1(b).

Essentially, at the sending side, the modem converts the digital signal output by the source digital device into an analog signal that is compatible with a normal speech signal. This is routed through the network in the same way as a speech signal and, at the receiving side, the modem converts the analog signal back again into its digital form before relaying this to the destination digital device. Modems also have the necessary circuits to set up and terminate a call. Hence by using a pair of modems – one at each subscriber access point – a PSTN can also be used to provide a switched digital service. The early modems supported only a very low bit rate service of 300 bps but, as a result of advances in digital signal processing circuits, modems are now available that support bit rates of up to 56 kbps. As we shall expand upon in Chapter 4, this is sufficient to support, not only applications that comprise text and images integrated together, but also services that comprise speech and low-resolution video.

In addition, continuing advances in digital signal processing techniques mean that modems are now available for use with the same access circuits that provide a high bit rate channel which is in addition to the speech channel used for telephony. Typically, the bit rate of this second channel is such that it can support high-resolution audio and video and hence they are used to provide access to servers that support a range of entertainment-related applications. The general scheme is shown in Figure 1.1(c) and, as we shall see in Chapter 4, such applications require bit rates in excess of 1.5 Mbps. This illustrates the technological advances that have been made in this area since the early modems were introduced in the early 1960s and, as we can deduce from this, a PSTN can now support not only speech applications but also a wide range of other multimedia communication applications.

1.3.2 Data networks

Data networks were designed to provide basic data communication services such as **electronic mail (email)** and general file transfers. The user equipment connected to these networks, therefore, is a computer such as a PC, a workstation, or an email/file server. The two most widely deployed networks of this type are the **X.25 network** and the **Internet**. Because of its operational mode, however, the X.25 network is restricted to relatively low bit rate data applications and hence is unsuitable for most multimedia applications.

The Internet is made up of a vast collection of interconnected networks all of which operate using the same set of **communication protocols**. A communication protocol is an agreed set of rules that are adhered to by all communicating parties for the exchange of information. The rules define not only the sequence of messages that are exchanged between the communicating parties but also the syntax of these messages. Hence by using the same set of communication protocols, all the computers that are connected to the Internet can communicate freely with each other irrespective of their type or manufacturer. This is also the origin of the term "**open systems interconnection**". Figure 1.2 shows a selection of the different types of interconnected network.

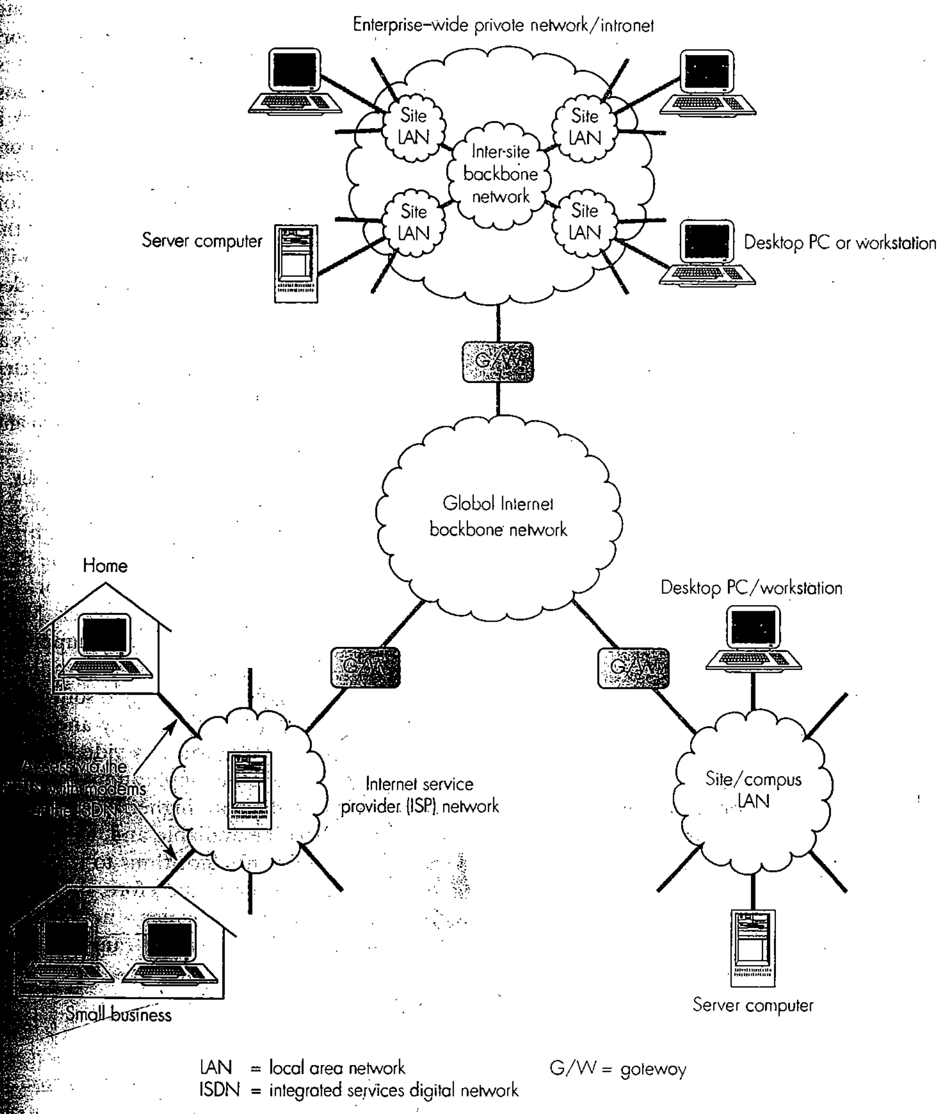

Enterprise-wide private network/intranet
Site LAN
Site LAN
Inter-site backbone network
Site LAN
Site LAN
Server computer
Desktop PC or workstation
G/W
Global Internet backbone network
Home
Desktop PC/workstation
G/W
G/W
Access via the PSTN with modems or the ISDN
Internet service provider (ISP) network
Site/campus LAN
Small business
Server computer
LAN = local area network
ISDN = integrated services digital network
G/W = gateway

Figure 1.2 A selection of the network types connected to the Internet.

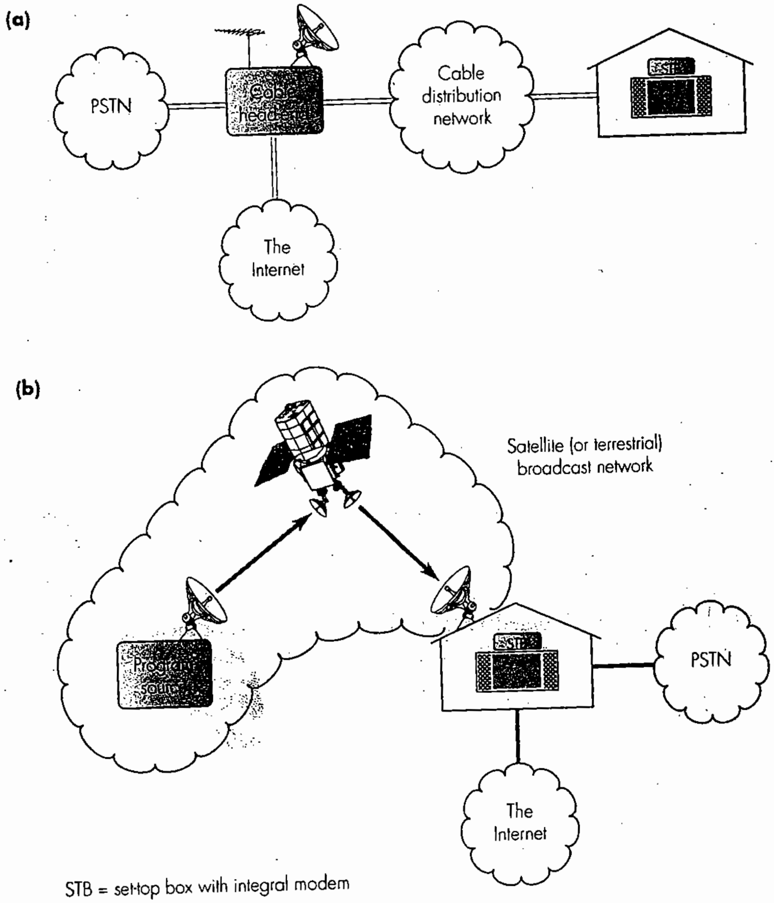
As we can see, in the case of a user at home or in a small business, access to the Internet is through an intermediate **Internet service provider (ISP) network**. Normally, since this type of user wants access to the Internet intermittently, the user devices are connected to the ISP network either through a PSTN with modems or through an **integrated services digital network (ISDN)** which, as we shall explain in Section 1.3.4, provides access at a higher **bit rate**. Alternatively, business users obtain access either through a **site/campus network** if the business comprises only a single site or, if it comprises multiple sites, through an **enterprise-wide private network**. The same approach is used by most colleges and universities. In the case of a single site/campus, the network is known as a (private) **local area network** or **LAN**. For an enterprise-wide network comprising multiple sites the sites are interconnected together using an **inter-site backbone network** to provide a set of enterprise-wide communication services. In addition, providing the communication protocols used by all the computers connected to the network are the same as those defined for use with the Internet, then all the users also have access to the range of services provided by the Internet. The enterprise network is then known as an **intranet** since all internal services are provided using the same set of communication protocols as those defined for the Internet. The different types of network are all connected to the **Internet backbone network** through an interworking unit called a **gateway** which, because it is responsible for routing and relaying all messages to and from the connected network, is also known as a **router**.

All data networks operate in what is called a **packet mode**. Essentially, a **packet** is a container for a block of data and, at its head, is the address of the intended recipient computer which is used to route the packet through the network. This mode of operation was chosen since the format of the data associated with data applications is normally in the form of discrete blocks of text or binary data with varying time intervals between each block. More recently, however, multimedia PCs have become available that support a range of other applications. For example, with the addition of a microphone and a pair of speakers – together with a sound card and associated software to digitize the speech – PCs are now used to support telephony and other speech-related applications. Similarly, with the addition of a video camera and associated hardware and software, a range of other applications involving video can be supported. Also, since their introduction, higher bit rate transmission circuits and routing nodes have become available and, as we shall expand upon in Chapters 3 and 4, more efficient algorithms to represent speech, audio, and video in a digital form. Collectively, therefore, this means that packet-mode networks – and the Internet in particular – now support not only general data communication applications but also a range of other multimedia communication applications involving speech, audio, and video.

1.3.3 Broadcast television networks

Broadcast television networks were designed to support the diffusion of analog television (and radio) programs throughout wide geographical areas. In the case of a large town or city, the broadcast medium is normally a **cable distribution network** while for larger areas, a **satellite network** or sometimes a **terrestrial broadcast network** is used. Since their introduction, digital television services have become available with these networks which, together with a low bit rate return channel for interaction purposes, provide a range of additional services such as games playing and home shopping. The general architecture of a cable distribution network and a satellite/terrestrial broadcast network are shown in Figure 1.3(a) and (b) respectively.

As we can see in Figure 1.3(a), the set-top box attached to the cable distribution network provides not only control of the television channels that are received but also access to other services. For example, when a **cable modem**



STB = set-top box with integral modem

Figure 1.3 Broadcast television networks: (a) cable networks; (b) satellite/terrestrial broadcast networks.

is integrated into the STB this provides both a low bit rate channel and a high bit rate channel from the subscriber back to the **cable head-end**. Typically, the low bit rate channel is used to connect the subscriber to a PSTN and the high bit rate channel to connect the subscriber to the Internet. Hence in addition to providing basic broadcast radio and television services, cable distribution networks also provide access to the range of multimedia communication services that are available with both a PSTN and the Internet. Similarly, as we can see in Figure 1.3(b), in the case of satellite and terrestrial broadcast networks, when a high-speed PSTN modem is integrated into the STB this provides the subscriber with an interaction channel so enhancing the range of services these networks support. This is the origin of the term "**interactive television**".

1.3.4 Integrated services digital networks

Integrated services digital networks started to be deployed in the early 1980s and were originally designed to provide PSTN users with the capability of having additional services. This was achieved firstly, by converting the access circuits that connect user equipment to the network – a telephone for example – into an all-digital form and secondly, by providing two separate communication channels over these circuits. These allow users either to have two different telephone calls in progress simultaneously or two different calls such as a telephone call and a data call. With an ISDN, therefore, the access circuit is known as a **digital subscriber line** (**DSL**).

The subscriber telephone can be either a digital phone or a conventional analog one. In the case of a digital phone, the electronics that are needed to convert the analog voice and call setup signals into a digital form are integrated into the phone handset. With an analog phone, the same electronics are located in the network termination equipment so making the digital mode of operation of the network transparent to the subscriber phone.

As we shall describe in Section 2.5.1, the digitization of a telephone-quality analog speech signal produces a constant bit rate binary stream – normally referred to as a **bitstream** – of 64 kbps. Hence, the basic DSL of the ISDN – known as the **basic rate access** or **BRA** – supports two 64 kbps channels. These can either be used independently (as they were intended) or as a single combined 128 kbps channel. Because of the design of an ISDN, however, since the two channels were intended for two different calls, this requires two separate circuits to be set up through the switching network independently. Hence to synchronize the two separate 64 kbps bitstreams into a single 128 kbps stream requires an additional box of electronics to perform, what is known as, the **aggregation** function.

In addition, a single higher bit rate channel of either 1.5 or 2 Mbps is supported. This is known as the **primary rate access** or **PRA**. Also, a more flexible way of obtaining a switched 128 kbps service has been introduced by many network operators. Indeed, the service provided has been enhanced and now supports a single switched channel of $p \times 64$ kbps where $p = 1, 2, 3 \ldots 30$. The various services provided are summarized in Figure 1.4 and, as we can deduce from this, an ISDN can support a range of multimedia applications. It should be noted, however, that because of the relatively high cost of digitizing the access circuits, in general the cost of the services associated with an ISDN are higher than the equivalent service provided by a PSTN.

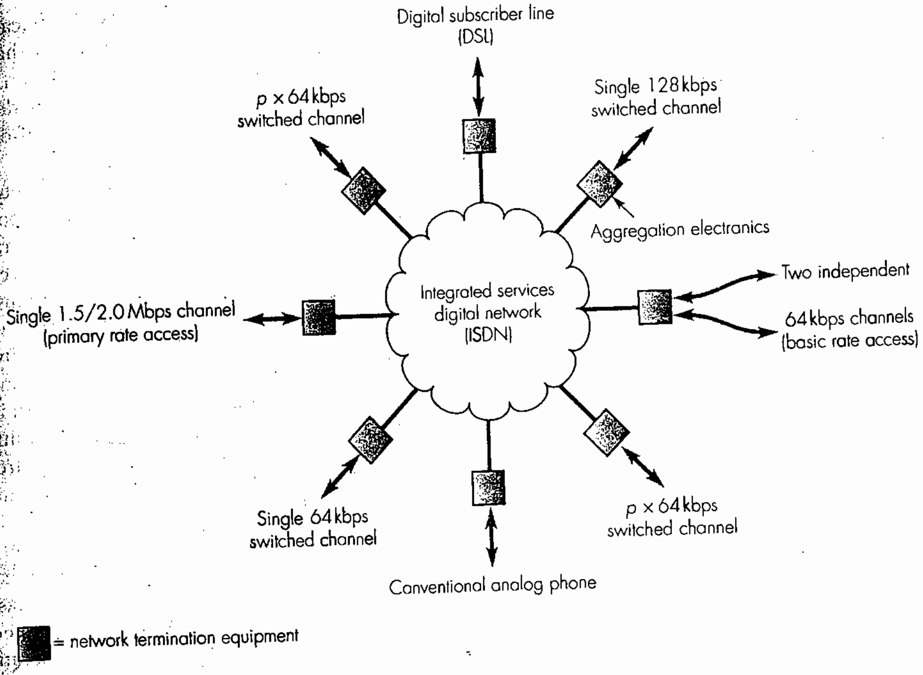

Digital subscriber line (DSL)
p × 64 kbps switched channel
Single 128 kbps switched channel
Aggregation electronics
Two independent 64 kbps channels (basic rate access)
Single 1.5/2.0 Mbps channel (primary rate access)
Integrated services digital network (ISDN)
Single 64 kbps switched channel
p × 64 kbps switched channel
Conventional analog phone
■ = network termination equipment

Figure 1.4 Alternative services provided by an ISDN.

1.3.5 Broadband multiservice networks

Broadband multiservice networks were designed in the mid-1980s for use as public switched networks to support a wide range of multimedia communication applications. The term "**broadband**" was used to indicate that the circuits associated with a call could have bit rates in excess of the maximum bit rate of 2 Mbps – 30×64 kbps – provided by an ISDN. As such, they were designed to be an enhanced ISDN and hence were called **broadband integrated services digital networks** or **B-ISDN**. Also, for the same reason, an ISDN is sometimes referred to as **narrowband ISDN** or **N-ISDN**.

At the time the B-ISDN was first conceived, the technology associated with the digitization of a video signal was such that, in general, an ISDN could not support services that included video. Since that time, however, considerable advances have been made in the field of compression with the effect that not only can an ISDN now support multimedia communication applications that include video, but also so can the other three types of network that we have described. The combined effect, therefore, has been to slow down considerably the deployment of B-ISDN. However, a number of the basic design features associated with the B-ISDN have been used as the basis of other broadband multiservice networks.

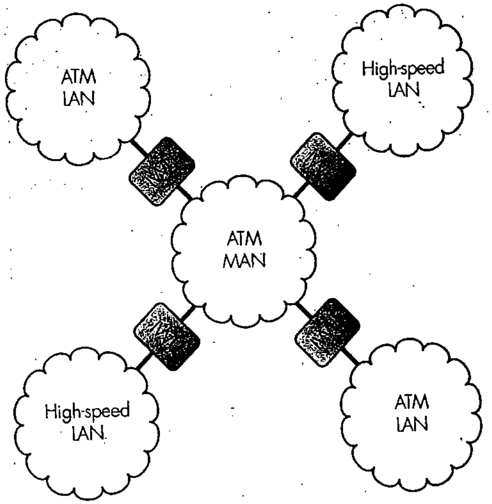
For example, by definition, multiservice networks implies that the network must support multiple services. In practice, however, as we shall expand upon in the next section, different multimedia applications require different bit rates, the rate being determined by the types of media that are involved. Hence the switching and transmission methods that are used within these networks must be more flexible than those used in networks such as a PSTN or ISDN which were initially designed to provide a single type of service. To achieve this flexibility, all the different media types associated with a particular multimedia application are first converted in the source equipment into a digital form. These are then integrated together and the resulting binary stream is divided into multiple fixed-sized packets known as **cells**. In practice, this type of information stream provides a much more flexible way of both transmitting and switching the multimedia information associated with the different types of application.

For example, in terms of transmission, the cells relating to the different applications can be integrated together more flexibly. Also, the use of fixed-sized cells means that the switching of cells can be carried out much faster than if variable-length packets were used. Since the different multimedia applications generate cell streams of different rates, this mode of operation means that the rate of transfer of cells through the network also varies and hence this mode of transmission is known as the **asynchronous transfer mode** or **ATM**. Broadband multiservice networks, therefore, are also known as **ATM networks** or sometimes **cell-switching networks**. For example, there are **ATM local area networks** (**ATM LANs**) that span a single site and **ATM metropolitan area networks** (**ATM MANs**) that span a large town or city. An example of a broadband multiservice network is shown in Figure 1.5 and, as we can see, the ATM MAN is being used as a high-speed backbone network to interconnect a number of LANs distributed around a large town or city. Note also that although two of the LANs are ATM LANs, the other two are simply higher-speed versions of older data-only LANs. This is typical of ATM networks in general which must often interwork with older (legacy) networks.

1.4 Multimedia applications

There are many and varied applications that involve multiple media types. In general, however, they can be placed into one of three categories:

- interpersonal communications,
- interactive applications over the Internet,
- entertainment applications.



ATM = asynchronous transfer mode LAN = local area network
MAN = metropolitan area network
IWU = interworking unit

Figure 1.5 Example of an ATM broadband multiservice network.

We shall discuss some examples from each category in the following sections. As we described in Section 1.1, however, in many instances the networks that are used to support these applications were initially designed to provide a service that involved just a single type of medium and it is as a result of advances in various related technologies that they are now used to support multimedia applications. Hence, in addition to these new services, the same networks are still used to support the basic application for which they were designed. Indeed, in some instances, the particular application supported has been enhanced. Thus, although from an applications perspective multimedia communications implies that two or more media types are involved, we shall also include selected examples of applications that these networks were designed to support even though only a single type of medium is involved.

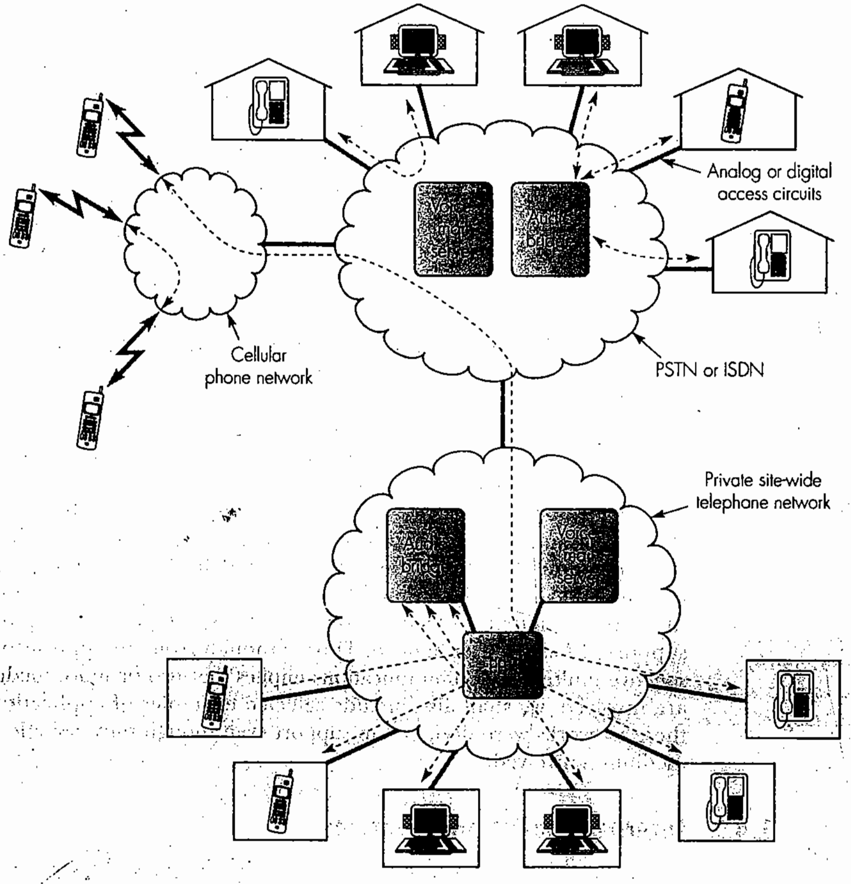
1.4.1 Interpersonal communications

Interpersonal communications may involve speech, image, text, or video. In some cases just a single type of medium is involved while in others two or more media types are integrated together. We shall discuss some examples from each category separately.

Speech only

Traditionally, interpersonal communications involving speech – telephony – have been provided using telephones that are connected either to a public switched telephone network (PSTN/ISDN/cellular phone network) or a PBX. The general scheme is shown in Figure 1.6.

Alternatively, by using a multimedia PC equipped with a microphone and speakers, the user can take part in telephone calls through the PC. This


Analog or digital access circuits
Cellular phone network
PSTN or ISDN
Private site-wide telephone network

PSTN = Public switched telephone network
PBX = Private branch exchange
ISDN = Integrated services digital netwarrk

gure 1.6 Speech-only interpersonal communications: public and private switched lephone networks.

requires a telephone interface card and associated software and is known a **computer telephony integration** or **CTI**. The added advantages of using a P instead of a conventional telephone are many. For example, the user ca create his or her own private directory of numbers and initiate a call simpl by selecting the desired number on the PC screen. More generally, providin the access circuit to the network has sufficient capacity – normally referre to as the circuit's **bandwidth** – it is possible to integrate telephony with all th other networked services provided by the PC.

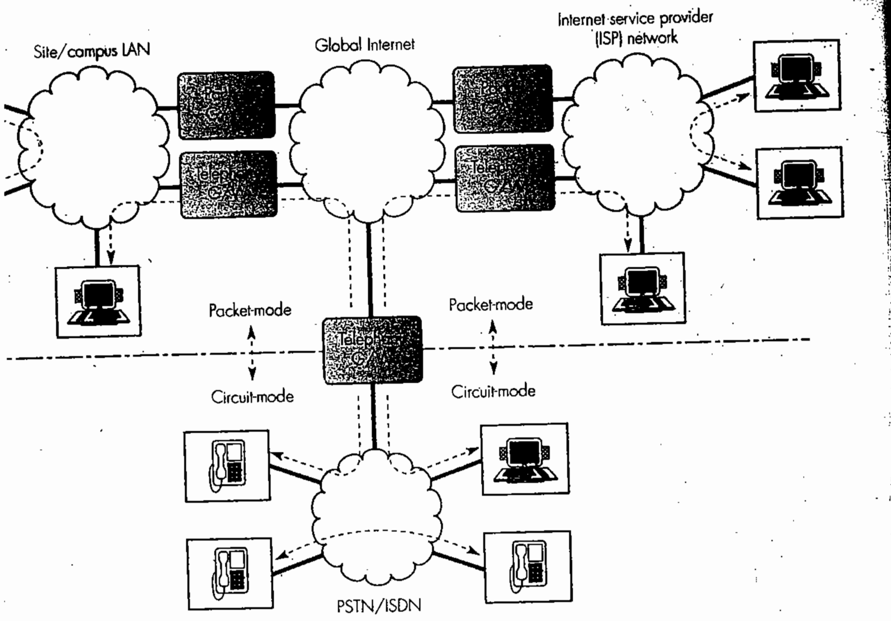
In addition to telephony, many public and private networks support additional services. Two examples are voice-mail and teleconferencing **Voice-mail**, for example, is used in the event of the called party being unavailable. A spoken message can then be left in the **voice mailbox** of the calle party. This is located in a central repository known as the **voice-mail server** The message can be read by the owner of the mailbox the next time he or she contacts the server.

Teleconferencing calls involve multiple interconnected telephones/PCs. Each person can hear and talk to all of the others involved in the call. This type of call is known variously as a **conference call** or, since it involves a telephone network, a **teleconferencing call** or sometimes an **audioconferencing call**. It requires a central unit known as an **audio bridge** which provides the necessary support to set up a conference call automatically.

The Internet is also used to support telephony. Initially, because the Internet was designed to support computer-to-computer communications, just (multimedia) PC-to-PC telephony was supported. This was subsequently extended so that a standard telephone could also be used. See Figure 1.7.

In the case of a PC-to-PC telephone call, the standard addresses that are used to identify individual computers connected to the Internet are used in the same way as for a data transfer application. However, because the Internet operates in a packet mode, both PCs must have the necessary hardware and software to convert the speech signal from the microphone into packets on input and back again prior to output to the speakers. Telephony over the Internet is also known, therefore, as **packet voice** or, because the network protocol associated with the Internet is called the Internet protocol (IP), **voice over IP (VoIP)**.

When a PC connected to the Internet needs to make a call to a telephone that is connected to a PSTN/ISDN, because these both operate in a circuit mode, an interworking unit known as a **telephony gateway** must be used. The PC user first sends a request to make a (telephone) call to a preallocated telephony gateway using the latter's Internet address. Then, assuming the user is registered to use this service, the gateway requests from the source PC the telephone number of the called party. On receipt of this, the source gateway initiates a session (call) with the telephony gateway nearest to the called party using the Internet address of the gateway. The called gateway then initiates a call to the recipient telephone using its telephone number and the standard call setup procedure of the PSTN/ISDN. Assuming the called party answers,



ay

Telephony over the Internet.

the called gateway then signals back to the PC user – through the source gateway – that the call can commence. A similar procedure is followed to clear the call on completion.

Image only

An alternative form of interpersonal communications over a PSTN or an ISDN is by the exchange of electronic images of documents. This type of service is known as **facsimile** – or simply **fax** – and is illustrated in Figure 1.8. Normally, this type of communication involves the use of a pair of fax machines, one at each network termination point. To send a document, the caller keys in the (telephone) number of the intended recipient and a circuit is set up through the network in the same way as for a telephone call. The two fax machines communicate with each other to establish operational parameters after which the sending machine starts to scan and digitize each page of the document in turn. Both fax machines have an integral modem within them and, as each page is scanned, its digitized image is simultaneously transmitted over the network and, as this is received at the called side, a printed version of the document is produced. Finally, after the last page of the document has been sent and received, the connection through the network is cleared by the calling machine in the normal way.

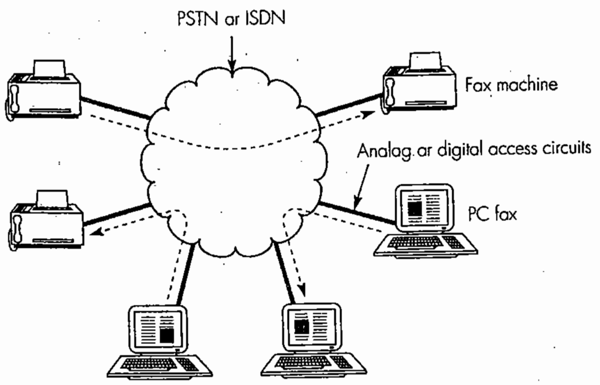


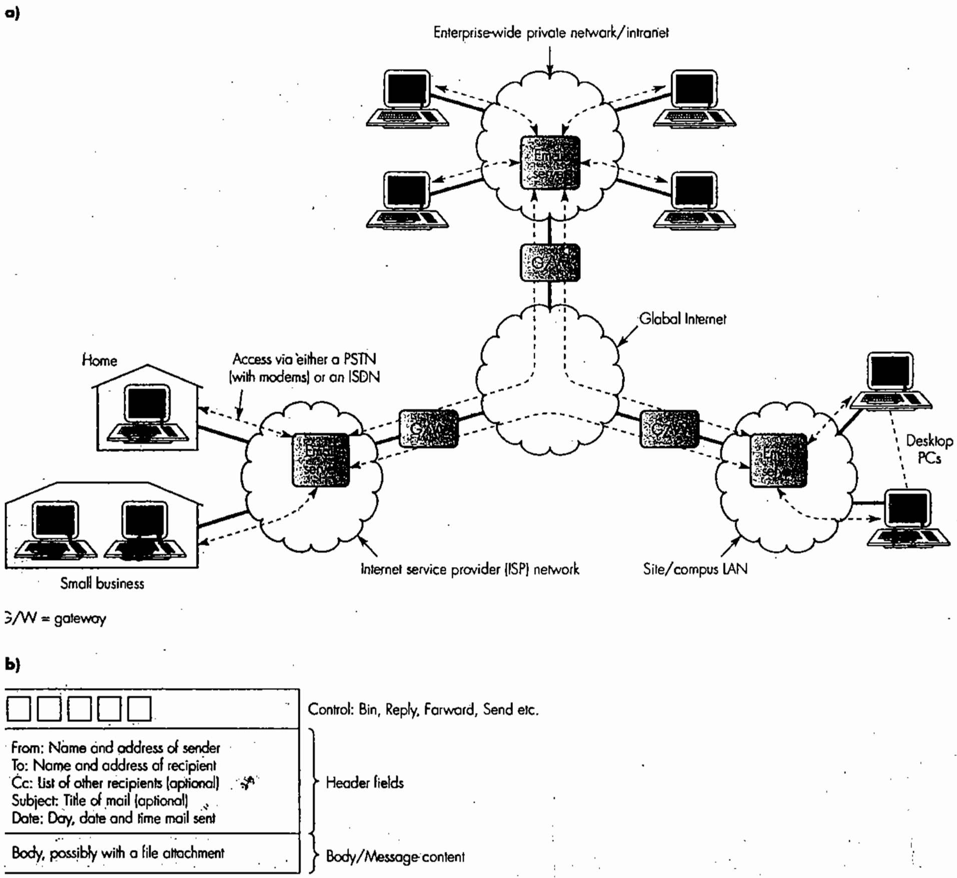
Figure 1.8 Image-only interpersonal communications: facsimile (fax) examples.

It is also possible to use a PC instead of a normal fax machine to send an electronic version of a document that is stored directly within the PC's memory. This mode of operation is known as **PC fax**. Essentially, the digital image of each page of the document is sent in the same way as the scanned image produced by a conventional fax machine. As with telephony, this requires a telephone interface card and associated software. The latter operates in exactly the same way as that in a fax machine and hence the terminal at the called side can be either a fax machine or another similar PC. In addition, with PC fax it is possible to send the digitized document over other network types such as an enterprise network. In this case, a LAN interface card and associated software are used. This mode of operation is particularly useful when working with paper-based documents such as invoices, and so on.

Text only

An example of interpersonal communications involving just text is electronic mail (email). The user terminal is normally a PC or a workstation and, as we described earlier in Section 1.3.2, the most widespread network used is the Internet. Various operational scenarios are shown in Figure 1.9(a).

In the case of a user at home, access to the Internet is through a PSTN/ISDN and an intermediate Internet service provider (ISP) network. Alternatively, business users obtain access either through an enterprise network or a site/campus network. Associated with each network is a set of one or more server computers. Each is known as an **email server** and, collectively, these contain a **mailbox** for each user connected to that network. A user can both create and deposit mail into his or her mailbox and read mail from it. Both the email servers and the internetwork gateway operate using the standard Internet communication protocols.


a)
Enterprise-wide private network/intranet
G/W
Global Internet
Home
Access via either a PSTN (with modems) or an ISDN
G/W
G/W
Desktop PCs
Small business
Internet service provider (ISP) network
Site/campus LAN
G/W = gateway
b)
Control: Bin, Reply, Forward, Send etc.
From: Name and address of sender
To: Name and address of recipient
Cc: List of other recipients (optional)
Subject: Title of mail (optional)
Date: Day, date and time mail sent
Header fields
Body, possibly with a file attachment
Body/Message content

igure 1.9 Text-only electronic mail: (a) email transfer examples; (b) example email essage format.

The format of a typical text-only email message is shown in Figure 1.9(b) and, as we can see, at the head is the unique Internet-wide name of both the sender and recipient of the mail. In addition, a copy of the mail can be sent to multiple recipients each of whom is listed in the cc part of the mail header, the acronym "cc" being the abbreviation for "carbon copy" which was the original means of making (paper) copies of documents. Normally, the contents of text-only mail comprise unformatted text, typically strings of ASCII characters.

Text and images

An example of an application that involves both text and images integrated together is **computer-supported cooperative working (CSCW)**. The network used is an enterprise network, a LAN, or the Internet and the general scheme is illustrated in Figure 1.10. Typically, a distributed group of people – each in his or her place of work – are all working on the same project. The user terminal is either a PC or a workstation and a window on each person's display is used as a shared workspace. This is known as a **shared whiteboard** and, normally, the display comprises text and images integrated together. The software associated with CSCW comprises a central program – known as the **whiteboard program** – and a linked set of support programs, one in each PC/workstation. The latter is made up of two parts: a change-notification part and an update-control part. Whenever a member of the group updates the contents of his or her whiteboard, the **change-notification** part sends details of the changes to the whiteboard program. This relays the changes to the **update-control** in each of the other PCs/workstations and these in turn proceed to update the contents of their copy of the whiteboard.

Speech and video

An example application that uses speech and video integrated together is **video telephony** which is now supported by all the network types. Figure 1.11(a) shows the general scheme.

In the case of the home, the terminals used are normally dedicated to providing the videophone service, while in an office, a single multimedia

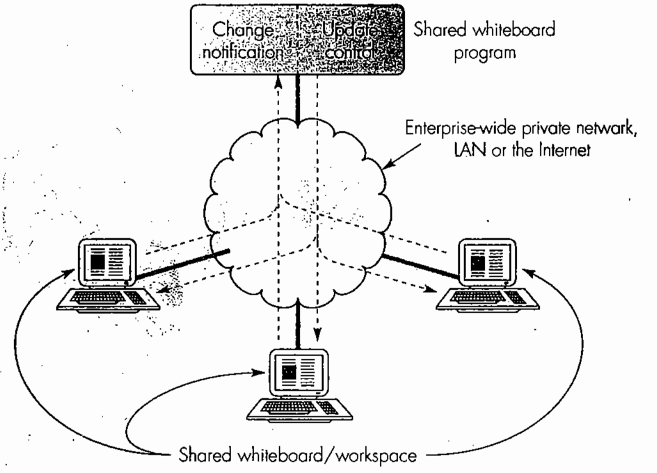

Change notification
Update control
Shared whiteboard program
Enterprise-wide private network, LAN or the Internet
Shared whiteboard/workspace

Figure 1.10 Text-and-image computer-supported cooperative working (CSCW).

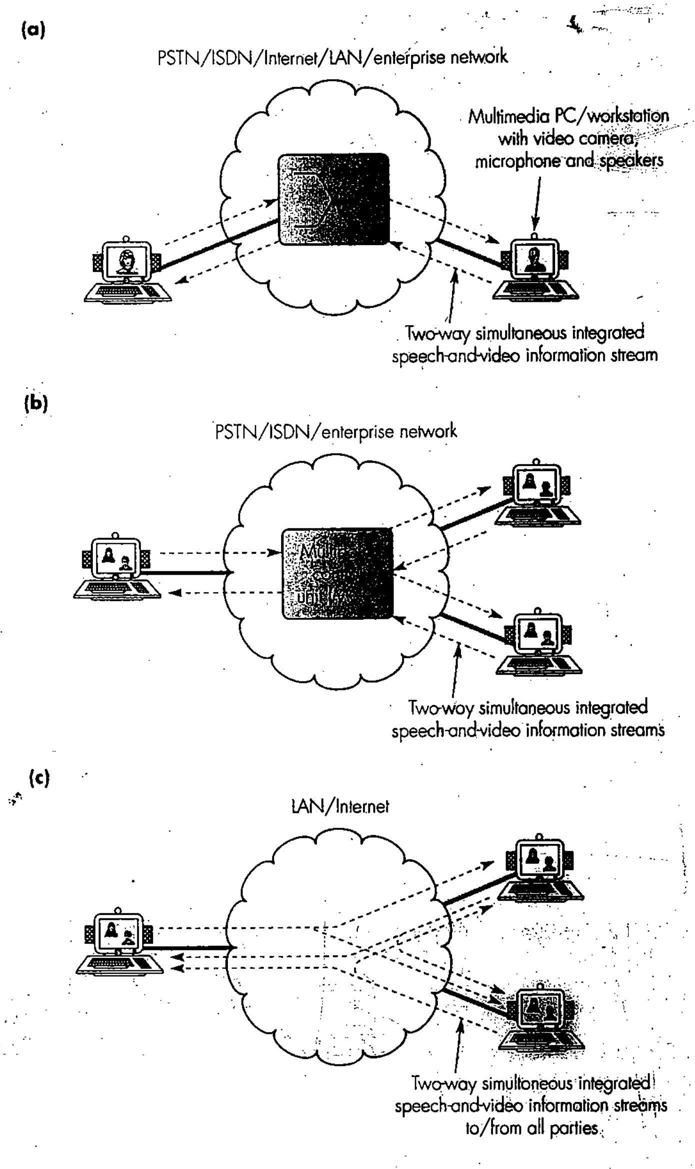

(a)
PSTN/ISDN/Internet/LAN/enterprise network
Multimedia PC/workstation with video camera, microphone and speakers
Two-way simultaneous integrated speech-and-video information stream
(b)
PSTN/ISDN/enterprise network
Two-way simultaneous integrated speech-and-video information streams
(c)
LAN/Internet
Two-way simultaneous integrated speech-and-video information streams to/from all parties

Figure 1.11 Speech-and-video interpersonal communications: (a) two-party video telephone call; (b) videoconferencing using an MCU; (c) videoconferencing using a broadcast network.

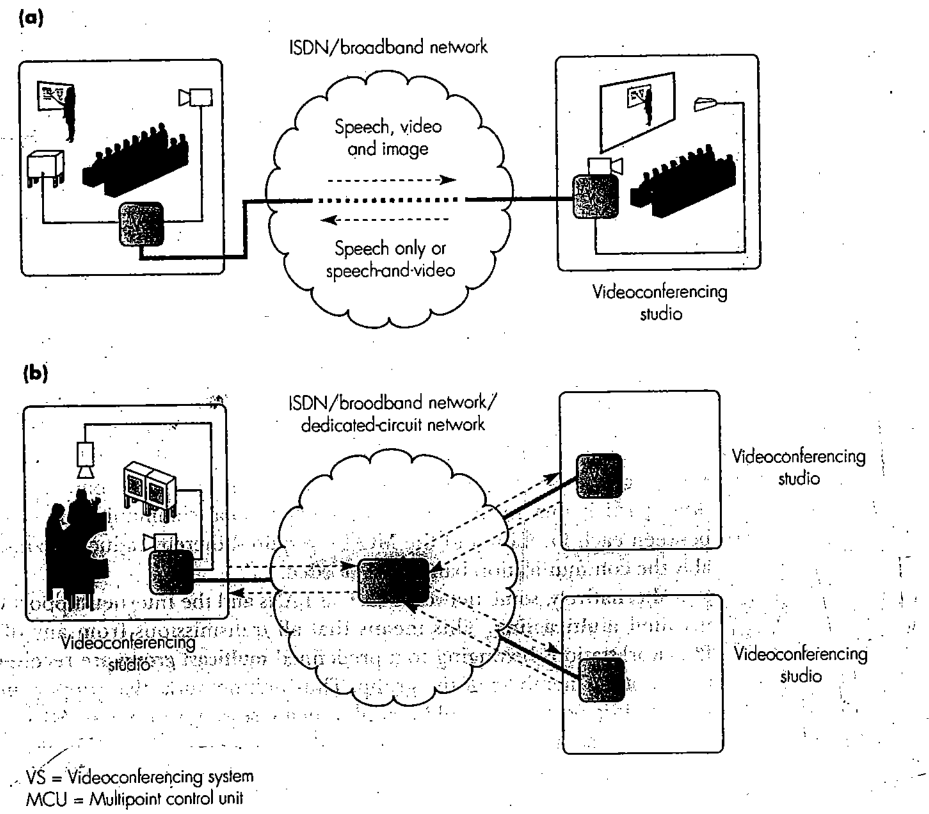
PC/workstation is used to provide the videophone service together with a range of other services. In both cases, the terminals/PCs incorporate a video camera in addition to the microphone and speaker used for telephony. With a dedicated terminal, a separate screen is used for the display whilst with a multimedia PC or workstation, the (moving) image of the called party is displayed in a window of the PC/workstation screen. The network must provide a two-way communication channel between the two parties of sufficient bandwidth to support the integrated speech-and-video generated by each terminal/PC.

The integration of video with speech means that the bandwidth of the access circuits required to support this type of service is higher than that required for speech only. Moreover, as with telephony, a call may involve not just two persons – and hence terminals/PCs – but several people each located in their own office. This type of call is then known as a **desktop videoconferencing call** and is now widely used in large corporations involving multiple geographically distributed sites in order to minimize travel between the various locations. As we indicated earlier, large corporations of this type have an enterprise-wide network to link the sites together and, in order to support videoconferencing, there is a central unit called a **multipoint control unit (MCU)** – or sometimes a **videoconferencing server** – associated with this network. An example is shown in Figure 1.11(b).

In principle, a separate window on the screen of each participant's PC/workstation should be used to display the video image of all the other participants. In practice, however, this would require multiple integrated speech-and-video communication channels, one for each participant, being sent to each of the other participants. Normally, this would require more bandwidth than is available. Hence instead, the integrated speech-and-video information stream from each participant is sent to the MCU which then selects just a single information stream to send to each participant. For example, with a voice-activated MCU, whenever the MCU detects a participan speaking, it relays the information stream from that participant to all the other participants. In this way, only a single two-way communication channe between each location and the MCU is required thereby reducing considerably the communication bandwidth needed.

Alternatively, some networks such as LANs and the Internet support wha is called **multicasting**. This means that all transmissions from any of th PCs/workstations belonging to a predefined **multicast group** are received b all the other members of the group. Thus with networks that support multicasting, it is possible to hold a conferencing session without an MCU. Th principle is shown in Figure 1.11(c) and, as we can deduce from this, this i only feasible when only a limited number of participants are involved owin to the high load it places on the network.

While the application just described involves only a single person at eac location, there are other applications that involve groups of people at one o more of the locations. Two examples are shown in Figure 1.12. In part (a) person at one location is communicating with a group of people at another location. This is the case, for example, with the transmission of a live lecture or seminar. Typically, the information stream transferred from the lecturer to the (remote) class would be integrated speech-and-video together with electronic copies of transparencies and other documents used in the lecture. In the reverse direction, the information may comprise just speech – for questions – or integrated speech-and-video to enable the lecturer to both see and hear the members of the class at the remote location. In terms of communications requirements, these are similar to those for a two-party videophone call. Alternatively, if the lecture is being relayed to multiple locations, either a separate communications channel is required to each remote site or an MCU



gure 1.12 Speech-and-video interpersonal communications: (a) remote lecture;) multiparty (group) videoconferencing.

is used at the lecturer's site. As we can see, because of the relatively high bandwidth that is involved, the network is either an ISDN that supports multiple 64 kbps channels or a broadband multiservice network if one is available.

In the example in Figure 1.12(b), there is a group of people at each location. This type of application has been in use for many years and was the first example of videoconferencing. Normally, since a group of people are present at each location, specially equipped rooms called **videoconferencing studios** are used which contain all the necessary audio and video equipment. This comprises one or more video cameras, a large-screen display, and associated audio equipment, all of which is connected to a unit called a **videoconferencing system**. A conference can involve just two locations or, more usually, multiple locations as shown in the figure. In the case of the latter, an MCU is normally used to minimize the bandwidth demands on the access circuits to the network. In the figure, the MCU is shown as a central facility within the network and hence only a single two-way communications channel is required for each access circuit of the network. This is the type of arrangement with a telecommunications-provider conference, for example. Alternatively, if a private network is being used, the MCU is normally located at one of the sites. The communication requirements at that site are then more demanding since it must support multiple input channels – one from each of the other sites – and a single output channel, the stream from which must be broadcast to all of the other sites.

Multimedia

The example discussed earlier concerning Internet-based electronic mail – **email** – assumed the information content of each email message consisted of text only. In addition, however, mail containing other media types such as images, audio, and video are also used. Three examples of electronic mail consisting of media types other than text are voice-mail, video-mail, and multimedia mail.

Voice-mail is similar in principle to that described earlier in relation to telephone networks. With Internet-based voice-mail, however, there is a voice-mail server associated with each network. This is in addition to the email server shown earlier in Figure 1.9(a). The user first enters a voice message addressed to the intended recipient and the local voice-mail server then relays this to the server associated with the intended recipient's network. The stored voice message is then played out the next time the recipient accesses his or her voice-mailbox. The same mode of operation is used for video-mail except in this case the mail message comprises an integrated speech-and-video sequence.

Multimedia mail is an extension of text-only mail inasmuch as the basic content of the mail comprises textual information. With multimedia mail, however, the textual information is annotated with a digitized image, a speech message, or a video message, as shown in Figure 1.13. In the case of speech-and-video, the annotations can be sent either directly to the mailbox

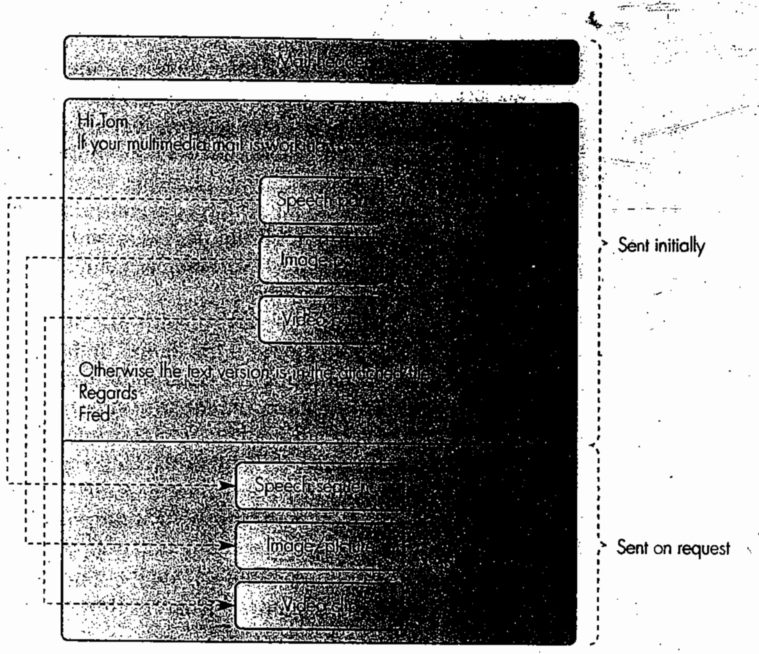

Mail header
Hi Tom,
If your multimedia mail is working
Speech
Image
Video
Otherwise the text version is in the attached
Regards
Fred
Speech segment
Image/picture
Video clip
Sent initially
Sent on request

Figure 1.13 Multimedia electronic mail structure.

of the intended recipient together with the original textual message – and hence stored and played out in the normal way – or they may have to be requested specifically by the recipient when the textual message is being read. In this way, the recipient can always receive the basic text-only message but the multimedia annotations can be received only if the terminal being used by the recipient supports voice and/or video.

1.4.2 Interactive applications over the Internet

In addition to a range of interpersonal communication applications, the Internet is also used to support a range of interactive applications, the most widely used being for interactions with a **World Wide Web (WWW)** or simply **Web**, server. This comprises a linked set of multimedia information servers that are geographically distributed around the Internet. The total information stored on all the servers is equivalent to a vast library of documents. The general principle is illustrated in Figure 1.14(a).

Each document comprises a linked set of **pages** and the linkages between the pages are known as **hyperlinks**. These are pointers – also known as **references** – either to other pages of the same document or to any other document within the total Web. In this way, a reader of a document has the

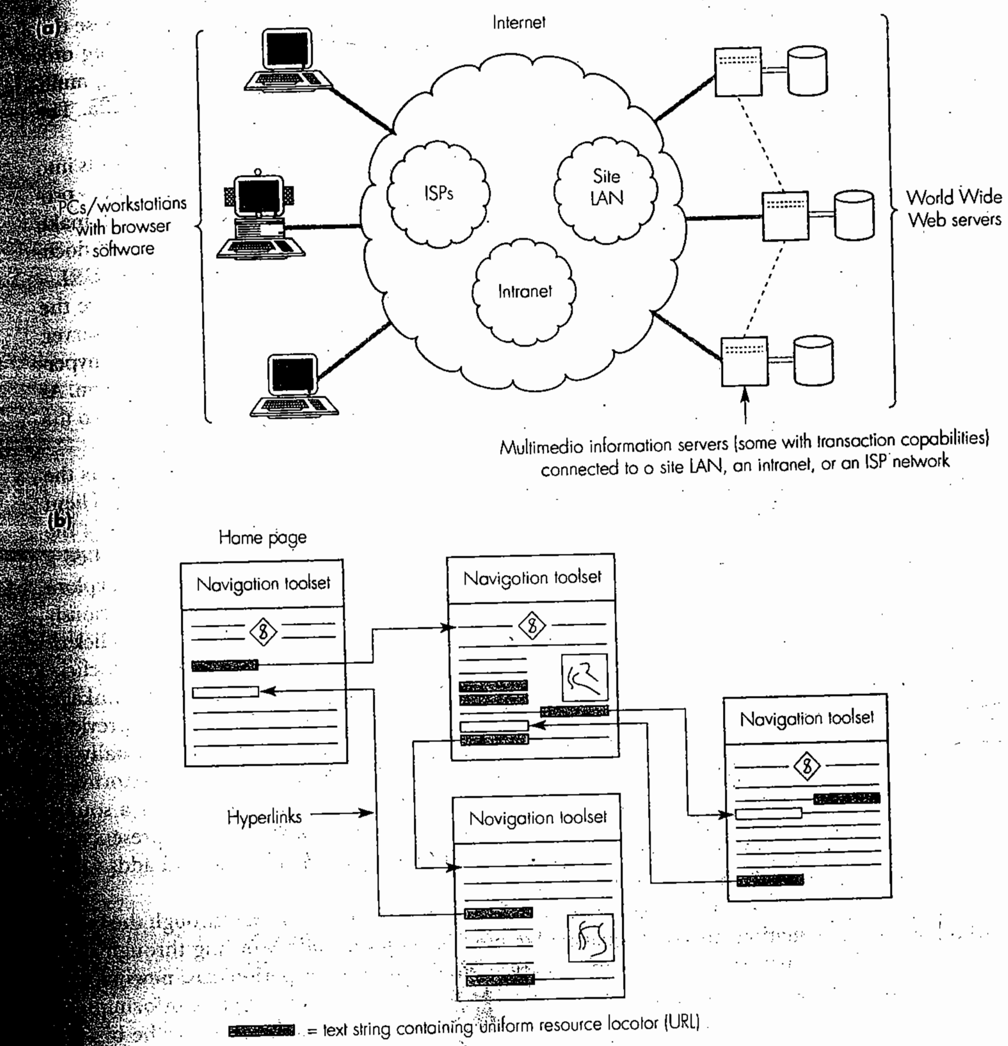

(a)
Internet
PCs/workstations with browser software
ISPs
Site LAN
Intranet
World Wide Web servers
Multimedia information servers (some with transaction capabilities) connected to a site LAN, an intranet, or an ISP network
(b)
Home page
Navigation toolset
Hyperlinks
▬ = text string containing uniform resource locator (URL)

Figure 1.14 Interactions with a World Wide Web server: (a) schematic; (b) hypertext linkages between the pages of a set of documents.

option, at well-defined points throughout the pages that make up a do
ment, to jump either to a different page of the same document or t
different document. Also, to return subsequently to a specific point on a p
at a later time. The optional linkage points within documents are defined

the creator of the document and are known as **anchors** since it is to these that the necessary linkage information is attached. Documents comprising only text are created using what is called **hypertext,** while those comprising multimedia information are created using what is known as **hypermedia**. The general structure of this type of document is shown in Figure 1.14(b).

There is no central authority for the introduction of new documents into the Web. Anyone can create a new document at a particular server site – providing the server has been allocated an Internet address – and make hyperlink references from it to any other document on the Web. Each document has a unique address – known as a **uniform resource locator** or **URL** – which identifies both the location of the server on the Internet where the first page of the document is stored and also the file reference on that server. The first page of a document is known as the **home page** and all the hyperlinks on this and the other pages have similar URLs associated with them. As we can deduce from this, the physical location of a page is transparent to the user and, in theory, can be located anywhere on the Web.

A standard format is used for writing documents. This is known as the **Hypertext Markup Language** (**HTML**) and it is also used for writing client software to explore the total contents of the Web, that is, the contents of the linked information on all the Web servers. The client function is called a **browser** and there are a number of user-friendly browsers available to explore the contents of the Web. These allow a user to create a directory of previously visited servers and to open up a dialog with a particular server at the click of the mouse. Once a desired document has been located, the user simply clicks on an anchor point within a page of the document to activate the linkage information stored at that point. It is also possible to return to the previous anchor at any time. With a hypertext document, the anchor is usually an underlined word or phrase while with a hypermedia document it is normally an icon of an appropriate shape; for example, a loudspeaker for a sound annotation or a video camera for a video clip. It is of course the presence of sound and video annotations that brings a document to life and adds value over a simple printed page.

In some applications the client simply wishes to browse through the information stored at a particular site. Examples include browsing through sales literature, product information, application notes periodicals, newspapers, and so on. In general, there is no charge for accessing this information. However, access to books, journals, and similar documents may be by subscription only.

In applications such as **homeshopping, homebanking,** and so on – more generally known as **teleshopping** and **telebanking** – a client may wish not only to browse through the information at a site but also to initiate an additional **transaction.** Here the server must provide additional transaction processing support for, say, ordering and purchasing. Since this will often involve a financial transaction, more rigorous security procedures are required for access and authentication purposes.

1.4.3 Entertainment applications

Entertainment applications can be one of two types:

- movie/video-on-demand,
- interactive television.

We shall discuss each application separately.

Movie/video-on-demand

This category of application is similar in principle to that described in the previous section except that, in general, the video and audio associated with entertainment applications must be of a much higher quality/resolution since wide-screen televisions and stereophonic sound are often used. As we shall describe in Section 4.3.3, a digitized movie/video – with sound – requires a minimum channel bit rate (bandwidth) of 1.5 Mbps. Hence the network used to support this type of application must be either a PSTN with a high bit rate modem – as we showed earlier in Figure 1.1(c) – or a cable network of the type we showed in Figure 1.3(a). As we saw earlier in Section 1.3.1, in the case of a PSTN, the high bit rate channel provided by the modem is used only over the access circuit and provides additional services to the other switched services that the PSTN supports. The general operational scheme in both cases is shown in Figure 1.15(a).

As we can see, the information stored on the server is a collection of digitized movies/videos. Normally the subscriber terminal comprises a conventional television with a selection device for interaction purposes. The user interactions are relayed to the server through a set-top box which also contains the high bit rate modem. By means of a suitable menu, the subscriber is able to browse through the set of movies/videos available and initiate the showing of a selected movie. This type of application is known as **movie-on-demand** (**MOD**) or sometimes **video-on-demand** (**VOD**). In addition to selecting a movie, the subscriber can control the showing of the movie by using similar controls to those used on a conventional video cassette recorder (VCR), that is, pause, fast-forward, and so on.

A key feature of MOD is that a subscriber can initiate the showing of a movie selected from a large library of movies at any time of the day or night. Hence, as we can deduce from Figure 1.15(b), this means that the server must be capable of playing out simultaneously a large number of video streams equal to the number of subscribers currently watching a movie. This requires the information flow from the server to be extremely high since it must support not just the transmission of a possibly large number of different movies, but also multiple copies of each movie. Technically this is very challenging and costly since, in general, the cost of the server is directly related to the aggregate information flow rate from it.

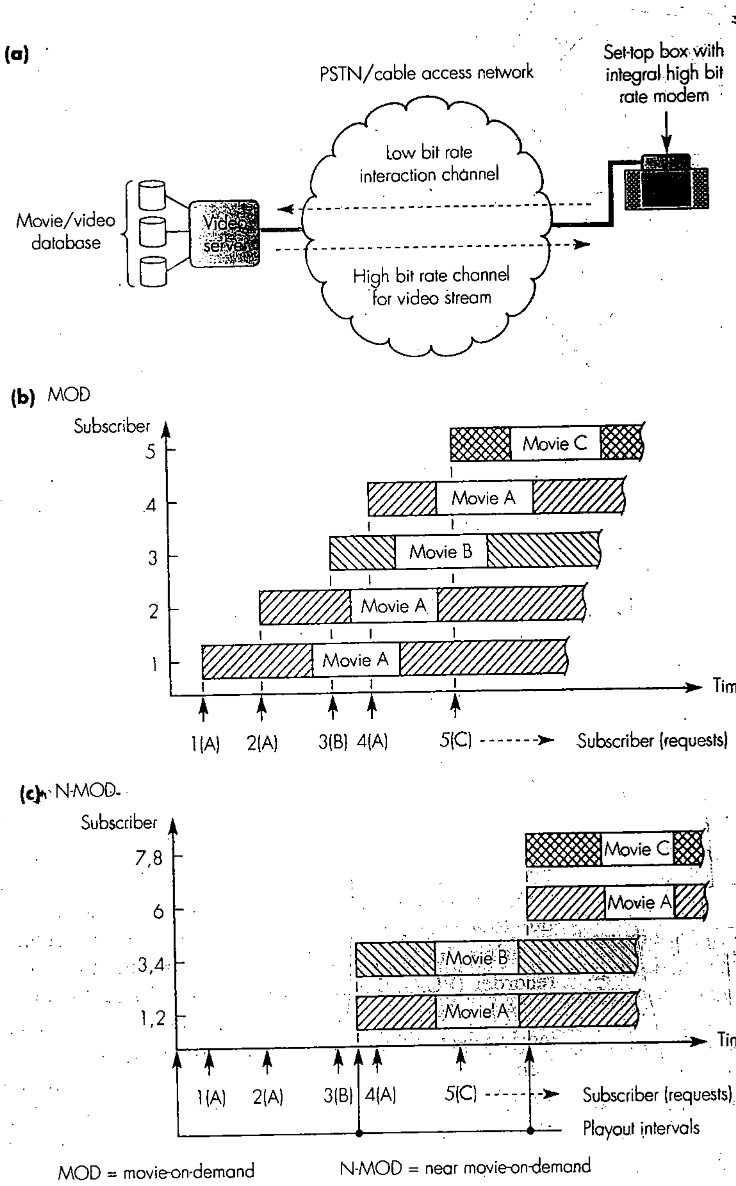

(a)
PSTN/cable access network
Set-top box with integral high bit rate modem
Low bit rate interaction channel
Movie/video database
Video server
High bit rate channel for video stream
(b) MOD
Subscriber
5
4
3
2
1
Movie C
Movie A
Movie B
Movie A
Movie A
Time
1(A) 2(A) 3(B) 4(A) 5(C)
Subscriber (requests)
(c) N-MOD
Subscriber
7,8
6
3,4
1,2
Movie C
Movie A
Movie B
Movie A
Time
1(A) 2(A) 3(B) 4(A) 5(C)
Subscriber (requests)
Playout intervals
MOD = movie-on-demand
N-MOD = near movie-on-demand

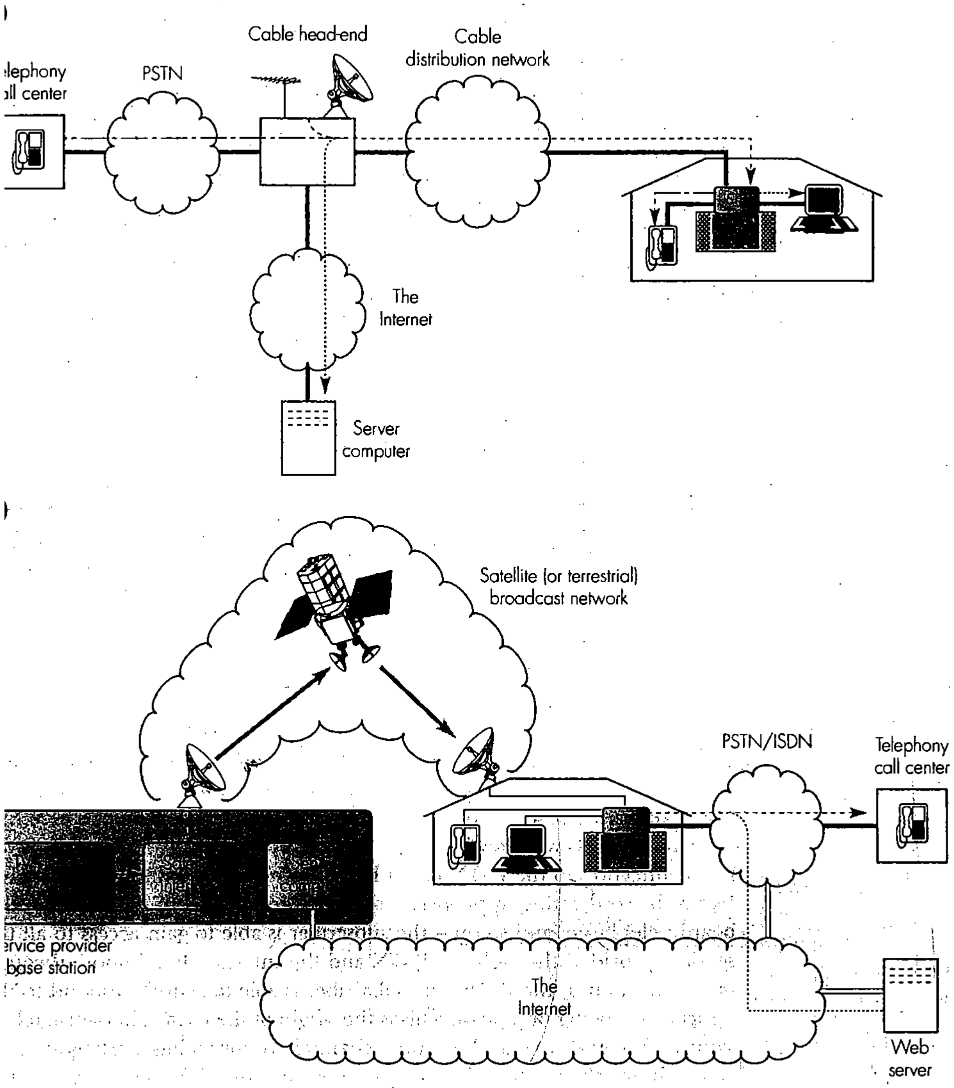
Figure 1.15 Interactions with a video server: (a) networking schematic; (b) movie-on-demand; (c) near movie-on-demand.

In practice, if the server is supporting a large number of subscribers, then it is common for several subscribers to request the same movie within a relatively short time interval between each request. An alternative mode of operation is also used, therefore, in which requests for a particular movie are not played out immediately but instead are queued until the start of the next playout time of that movie as shown in Figure 1.15(c). In this way, all requests for the same movie which are made during the period up to the next playout time are satisfied simultaneously by the server outputting a single video stream. This mode of operation, is known as **near movie-on-demand** or **N-MOD**. Clearly, however, the viewer is unable to control the playout of the movie.

Similar applications are also used within a business environment, except that the stored information in the server is typically training and general educational material, company news, and so on, and thus the number of stored videos is normally much less as is the number of simultaneous users. This means that the video servers required are less sophisticated than those used in public MOD/N-MOD systems. Also, the stored video streams/programs are often in a different format. The format used is the same as that used with **CD-ROMs** since the received video stream can then be displayed directly on the screen of a multimedia PC or workstation. The communication requirements of the private network, however, are the same as those identified for use with a public network.

Interactive television

Broadcast television networks include cable, satellite, and terrestrial networks. The basic service provided by these networks is, of course, the diffusion of both analog and digital television (and radio) programs. In addition, however, as we saw earlier in Section 1.3.3, the set-top box (STB) associated with these networks also has a modem within it. In the case of a cable network, as we show in Figure 1.16(a), the STB provides both a low bit rate connection to the PSTN and a high bit rate connection to the Internet. Hence by connecting appropriate terminal equipment to the STB – a keyboard, telephone, and so on – the subscriber is able to gain access to all the services provided through the PSTN and the Internet. In addition, through the connection to the PSTN, the subscriber is able to actively respond to the information being broadcast. This is the origin of the term "interactive television" and typical uses of the return channel are for voting, participation in games, home shopping and so on. As we see in Figure 1.16(b), a similar set of services are available through satellite and terrestrial broadcast networks, except that the STB associated with these networks requires a high-speed modem to provide the connections to the PSTN and the Internet.


lephony
ll center
PSTN
Cable head-end
Cable distribution network
The Internet
Server computer
Satellite (or terrestrial) broadcast network
PSTN/ISDN
Telephony call center
rvice provider
base station
The Internet
Web server

gure 1.16 Interactive television: (a) cable distribution network; (b) satellite/terrestrial oadcast network.

1.5 Application and networking terminology

Before we leave this chapter it will be helpful if we first review some of the terminology used in relation to the different media types and also the terminology and operational characteristics of the different types of communication channels provided by the various networks we have identified. A selection of the terms that are used are shown in Figure 1.17. We shall describe each term separately as well as its origin and interrelationship with the other terms.

1.5.1 Media types

As we identified in Section 1.4 when we described the different multimedia applications, the information flow associated with the different applications can be either continuous or block-mode. In the case of **continuous media**, this means that the information stream is generated by the source continuously in a time-dependent way. In general, therefore, continuous media is

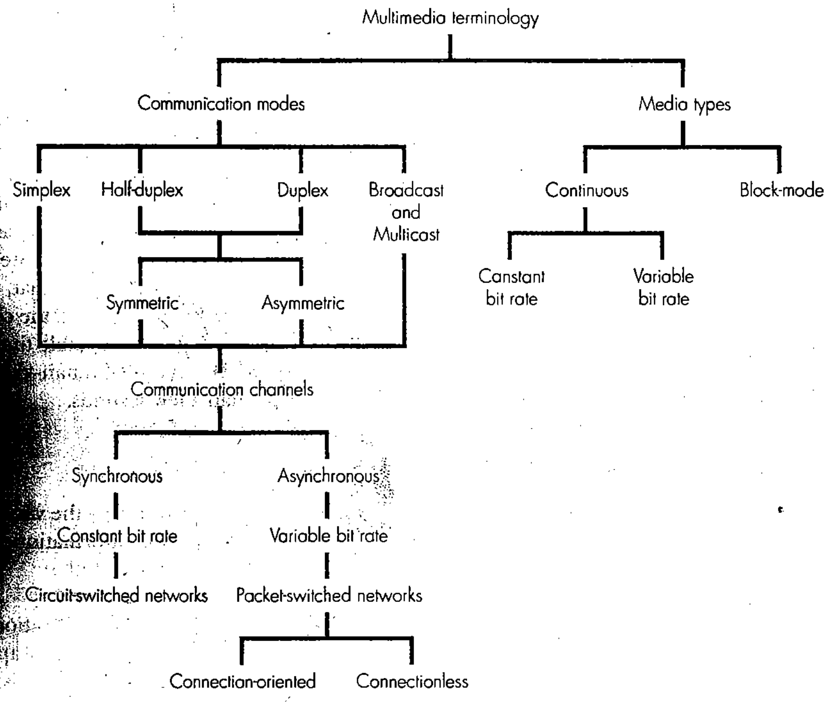

Multimedia terminology
Communication modes
Media types
Simplex
Half-duplex
Duplex
Broadcast and Multicast
Continuous
Block-mode
Symmetric
Asymmetric
Constant bit rate
Variable bit rate
Communication channels
Synchronous
Asynchronous
Constant bit rate
Variable bit rate
Circuit-switched networks
Packet-switched networks
Connection-oriented
Connectionless

Figure 1.17 A selection of the terms used with multimedia.

passed directly to the destination as it is generated and, at the destination, the information stream is played out directly as it is received. This mode of operation is called **streaming** and, since continuous media is generated in a time-dependent way, it is also known as **real-time media**. With continuous media, therefore, the bit rate of the communications channel that is used must be compatible with the rate the source media is being generated. Two examples of media types that generate continuous streams of information in real time are audio and video.

In terms of the bit rate at which the source information stream is generated, this may be at either a **constant bit rate (CBR)** or a **variable bit rate (VBR)**. With audio, for example, the digitized audio stream is generated at a constant bit rate which is determined by the frequency the audio waveform is sampled and the number of bits that are used to digitize each sample. In the case of video, however, as we shall expand upon in Section 2.6.1, although the individual pictures/frames that make up the video are generated at a constant rate, after compression, the amount of information associated with each frame varies. In general, therefore, the information stream associated with compressed video is generated at fixed time intervals but the resulting bit rate is variable.

In the case of **block-mode media**, the source information comprises a single block of information that is created in a time-independent way. For example, a block of text representing an email or computer program, a two-dimensional matrix of pixel values that represents an image, and so on. Normally, therefore, block-mode media is created in a time-independent way and is often stored at the source in, say, a file. Then, when it is requested, the block of information is transferred across the network to the destination where it is again stored and subsequently output/displayed at a time determined by the requesting application program. This mode of operation is known as **downloading** and, as we can deduce with block-mode media, the bit rate of the communications channel need not be constant but must be such that, when a block is requested, the delay between the request being made and the contents of the block being output at the destination is within an acceptable time interval. This is known as the **round-trip delay (RTD)** and, for human–computer interactions, ideally, this should be no more than a few seconds.

1.5.2 Communication modes

In terms of the communication channels that are provided by the various network types, as we show in Figure 1.18, the transfer of the information streams associated with an application can take place in one of five modes:

- **simplex:** this means the information associated with the application flows in one direction only. An example is the transmission of photographic images from a deep-space probe at predetermined times since this involves just a unidirectional flow of information from the probe to an earth station;

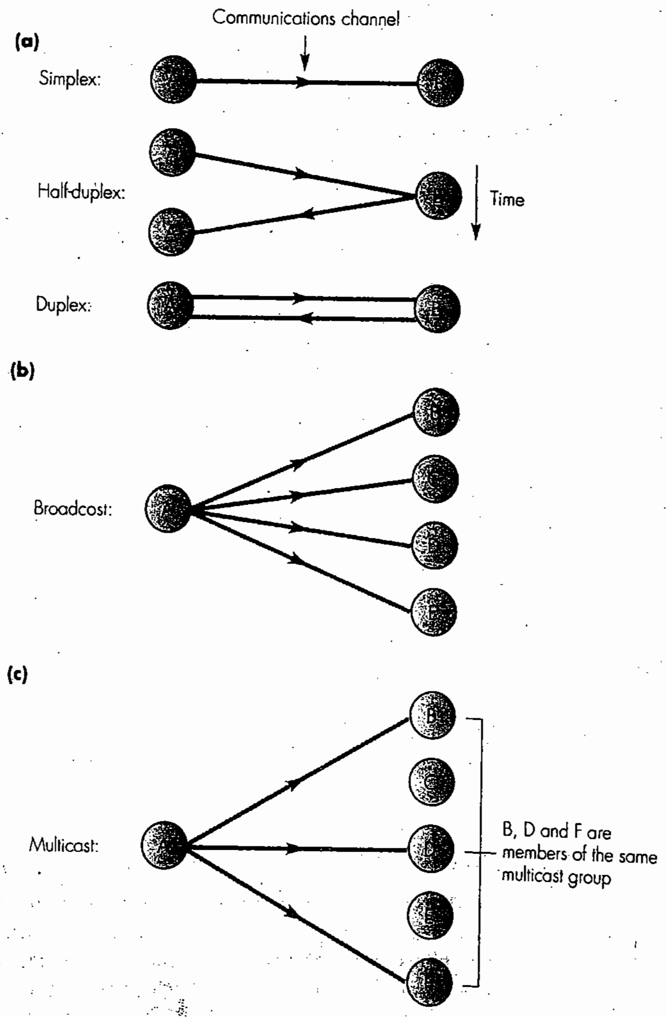


Figure 1.18 Communication modes: (a) unicast; (b) broadcast; (c) multicast.

- **half-duplex:** this means that information flows in both directions but alternately. This mode is also known as **two-way alternate** and an example is a user making a request for some information from a remote server and the latter returning the requested information;

- **duplex:** this means that information flows in both directions simultaneously. It is also known as **two-way simultaneous** and an example is the two-way flow of digitized speech and video associated with a video telephony application;
- **broadcast:** this means that the information output by a single source node is received by all the other nodes – computers, and so on – that are connected to the same network. An example is the broadcast of a television program over a cable network as all the television receivers that are connected to the network receive the same set of programs;
- **Multicast:** this is similar to a broadcast except that the information output by the source is received by only a specific subset of the nodes that are connected to the network. The latter form what is called a **multicast group** and an example application is videoconferencing which involves a predefined group of terminals/computers connected to a network exchanging integrated speech and video streams.

In the case of half-duplex and duplex communications, the bit rate associated with the flow of information in each direction can be either equal or different; if the flows are equal, the information flow is said to be **symmetric** and if the flows are different, **asymmetric**. For example, a video telephone call involves the exchange of an integrated digitized speech and video stream in both directions simultaneously and hence a symmetric duplex communications channel is required. Alternatively, in an application involving a browser (program) and a Web server, a low bit rate channel from the browser to the Web server is required for request and control purposes and a higher bit rate channel from the server to the subscriber for the transfer of, say, the requested file. Hence for this type of application, an asymmetric half-duplex communications channel is sufficient.

1.5.3 Network types

In the same way that there are two types of information stream associated with the different media types – continuous and block-mode – so there are two types of communications channel associated with the various network types, one that operates in a time-dependent way known as **circuit-mode** and the other in a time-varying way known as **packet-mode**. The first is known as a **synchronous communications channel** since it provides a constant bit rate service at a specified rate. The second is known as an **asynchronous communications channel** since it provides a variable bit rate service, the actual rate being determined by the (variable) transfer rate of packets across the network.

Circuit-mode

A circuit-mode network is shown in Figure 1.19 and, as we can see, it comprises an interconnected set of **switching offices/exchanges** to which the subscriber terminals/computers are connected. This type of network is known as a **circuit-switched network** and, prior to sending any information, the source must first set up a connection through the network. Each subscriber terminal/computer has a unique network-wide number/address associated with it and, to make a call, the source first enters the number/address of the intended communication partner. The local switching office/exchange then uses this to set up a connection through the network to the switching office/exchange to which the destination is connected and, assuming the destination is free and ready to receive a call, a message is returned to the source indicating that it can now start to transfer/exchange information. Finally, after all the information has been transferred/exchanged, either the source or the destination requests for the connection to be cleared. The bit rate associated with the connection is fixed and, in general, is determined by the bit rate that is used over the access circuits that connect the source and destination terminal/computer to the network.

The messages associated with the setting up and clearing of a connection are known as **signaling messages**. As we can deduce from the above, with a circuit-switched network there is a time delay while a connection is being established. This is known as the **call/connection setup delay** and two examples of networks that operate in this way are a PSTN and an ISDN. With

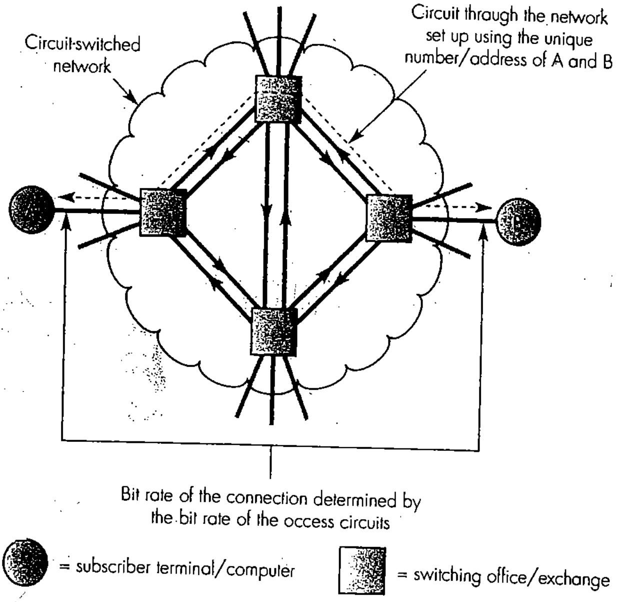

Circuit-switched network
Circuit through the network set up using the unique number/address of A and B
Bit rate of the connection determined by the bit rate of the access circuits
= subscriber terminal/computer
= switching office/exchange

Figure 1.19 Circuit-switched network schematic.

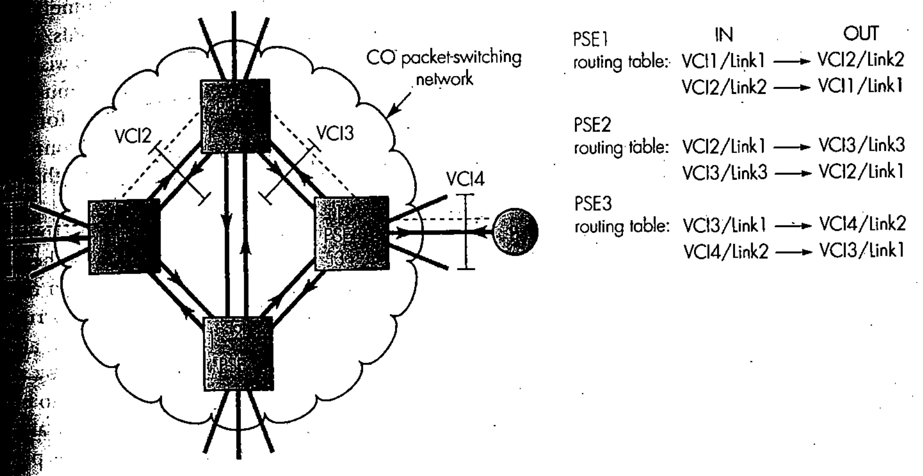
a PSTN, the call setup delay can range from a fraction of a second for a local call through to several seconds for an international call. With an ISDN, however, the delay ranges from tens of milliseconds through to several hundred milliseconds.

Packet-mode

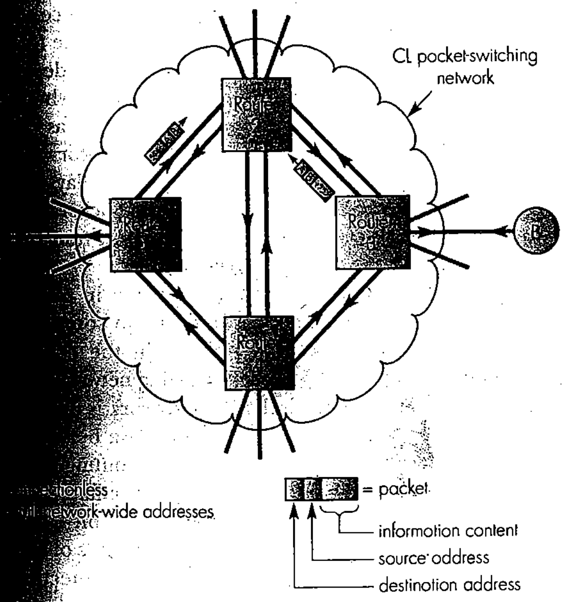
As we see in Figure 1.20, there are two types of packet-mode network: **connection-oriented** (**CO**) and **connectionless** (**CL**). The principle of operation of a connection-oriented network is shown in Figure 1.20(a) and, as we can see, it comprises an interconnected set of **packet-switching exchanges** (**PSEs**). This type of network is known as a **packet-switched network** and, as with a circuit-switched network, each terminal/computer that is connected to the network has a unique network-wide number/address associated with it. With a connection-oriented network, as the name implies, prior to sending any information, a connection is first set up through the network using the addresses of the source and destination terminals. However, in a packet-switched network, the connection/circuit that is set up utilizes only a variable portion of the bandwidth of each link and hence the connection is known as a **virtual connection** or, more usually, a **virtual circuit** (**VC**).

To set up a VC, the source terminal/computer sends a *call request* control packet to its local PSE which contains, in addition to the address of the source and destination terminal/computer, a short identifier known as a **virtual circuit identifier** (VCI) Each PSE maintains a table that specifies the outgoing link that should be used to reach each network address and, on receipt of the *call request* packet, the PSE uses the destination address within the packet to determine the outgoing link to be used. The next free identifier (VCI) for this link is then selected and two entries are made in a **routing table**. The first specifies the incoming link/VCI and the corresponding outgoing link/VCI and the second, in order to route packets in the reverse direction, the inverse of these, as we show in the example in the figure. The *call request* packet is then forwarded on the selected outgoing link and the same procedure is followed at each PSE along the route until the destination terminal/computer is reached.

Collectively, the VCIs that are used on the various links form the virtual circuit and, at the destination, assuming the call is accepted, a *call accepted* packet is returned to the source over the same route/virtual circuit. The information transfer phase can then start but, since a VC is now in place, only the VCI is needed in the packet header instead of the full network-wide address. Each PSE first uses the incoming link/VCI to determine the outgoing link/VCI from the routing table. The existing VCI in the packet header is then replaced with that obtained from the routing table and the packet is forwarded on the identified outgoing link. The same procedure is followed to return information in the reverse direction and, when all information has been transferred/exchanged, the VC is cleared and the appropriate VCIs are released by passing a *call clear* packet along the VC.


CO packet-switching network
VCI2
VCI3
VCI4
PSE1 IN OUT
routing table: VCI1/Link1 → VCI2/Link2
VCI2/Link2 → VCI1/Link1
PSE2
routing table: VCI2/Link1 → VCI3/Link3
VCI3/Link3 → VCI2/Link1
PSE3
routing table: VCI3/Link1 → VCI4/Link2
VCI4/Link2 → VCI3/Link1
VCI = virtual circuit identifier
PSE = packet-switching exchange

ection-oriented
circuit


CL packet-switching network
= packet
information content
source address
destination address

ionless
work-wide addresses

1.20 **Packet-switching network principles: (a) connection-oriented; (b) connectionless.**

In contrast, with a connectionless network, the establishment of a co nection is not required and the two communicating terminals/compute can communicate and exchange information as and when they wish. In ord to do this, however, as we show in Figure 1.20(b), each packet must carry t full source and destination addresses in its header in order for each PS to route the packet onto the appropriate outgoing link. In a connectionle network, therefore, the term **router** is normally used rather than packe switching exchange.

In both network types, as each packet is received by a PSE/router on a incoming link, it is stored in its entirety in a memory buffer. A check is the made to determine if any transmission/bit errors are present in the pack header – that is, the signal that is used to represent a binary 0 is corrupted an is interpreted by the receiver as a binary 1 and vice versa – and, if an error detected, the packet is simply discarded. The service offered by a packe switched network is said, therefore, to be a **best-effort service.** If no errors a detected then the addresses/VCIs carried in the packet header are read determine the outgoing link that should be used and the packet is placed in queue ready for forwarding on the selected outgoing link. All packets are tran mitted at the maximum link bit rate. However, with this mode of operation, it possible for a sequence of packets to be received on a number of incomin links all of which need forwarding on the same outgoing link. Hence a pack may experience an additional delay while it is in the output queue for a lin waiting to be transmitted. Clearly, this delay will be variable since it depends o the number of packets that are currently present in the queue when a ne packet arrives for forwarding. This mode of operation is known as (packe **store-and-forward** and, as we can see, there is a packet store-and-forward del in each PSE/router. The sum of the store-and-forward delays in ea PSE/router contributes to the overall transfer delay of the packet across t network. The mean of this delay is known as the **mean packet transfer del** and the variation about the mean the **delay variation** or **jitter.**

An example of a packet-switched network that operates in the conne tionless mode is the Internet, which we shall describe in some detail Chapter 9. Two examples of networks that operate in the connectio oriented mode are the international X.25 packet-switching network and AT networks. As we explained in Section 1.3.2, the X.25 network is used prim ily for the transfer of files containing text and binary data between lar computers. Because of the packet format that is used, the routing of pack is relatively slow with the effect that the X.25 network is unsuitable for m multimedia applications. In contrast, as we described in Section 1.3.5, AT networks have been designed from the outset to support all types of multim dia applications. This is achieved by using high bit rate interconnecting lin and, once a virtual circuit has been set up, a very small fixed-sized packet 53 bytes is used to transfer the information associated with the call. Eac small packet is known as a **cell** and includes a short 5-byte header whic enables cells to be switched at the very high link bit rates that are used. It

for this reason that ATM networks are also known as **fast packet-switching works** or sometimes **cell-switching networks**. We shall describe the opera of ATM networks in Chapter 10.

1.5.4 Multipoint conferencing

As we described in Section 1.4.1, multipoint conferencing features in m interpersonal applications including audio- and videoconferencing, sharing, and computer-supported cooperative working. Essentially, th involve the exchange of information between three or more terminals/c puters. In practice, because of the different modes of operation of the network types – circuit-switched and packet-switched – multipoint confer ing is implemented in one of two ways: centralized and decentralized.

The **centralized mode** is used with circuit-switched networks such PSTN or an ISDN and, as we show in Figure 1.21(a), with this mode a cen ized conference server is used. Prior to sending any information, e terminal/computer to be involved in the conference must first set up a nection to the server. Each terminal/computer then sends its own me stream – comprising, say, audio, video, and data integrated together in so way – to the server using the established connection. The server, in turn, t distributes either the media stream received from a selected terminal/c puter or a mix of the media streams received from several termin computers back to all the other terminals/computers that are involve the conference.

The **decentralized mode** is used with packet-switched networks that port multicast communications. Examples include local area netwo intranets, and the Internet. In this mode, as we show in Figure 1.21(b), output of each terminal/computer is received by all the other member the conference/multicast group. Hence a conference server is not norm used and instead it is the responsibility of each terminal/compute manage the information streams that it receives from the other members.

In addition, a third mode known as the **hybrid mode** can be used. Th shown in Figure 1.21(c) and, as we can see, it is used when the various te nals/computers that make up the conference are attached to diffe network types. In the example shown, the conference comprises four te nals/computers, two attached to a circuit-switched network and two packet-switched network that supports multicasting. As in the centrali mode, a conference server is used and the output of each terminal/c puter is sent to the server either over individual circuits – terminals A and or using multicasting – terminals C and D. However, in this mode, as in centralized mode, it is the server that determines the output stream(s) to sent to each terminal.

As we explained in Section 1.4.1, there are four types of conferencing:

- data conferencing: this involves data only and examples include data sharing and computer-supported cooperative working;

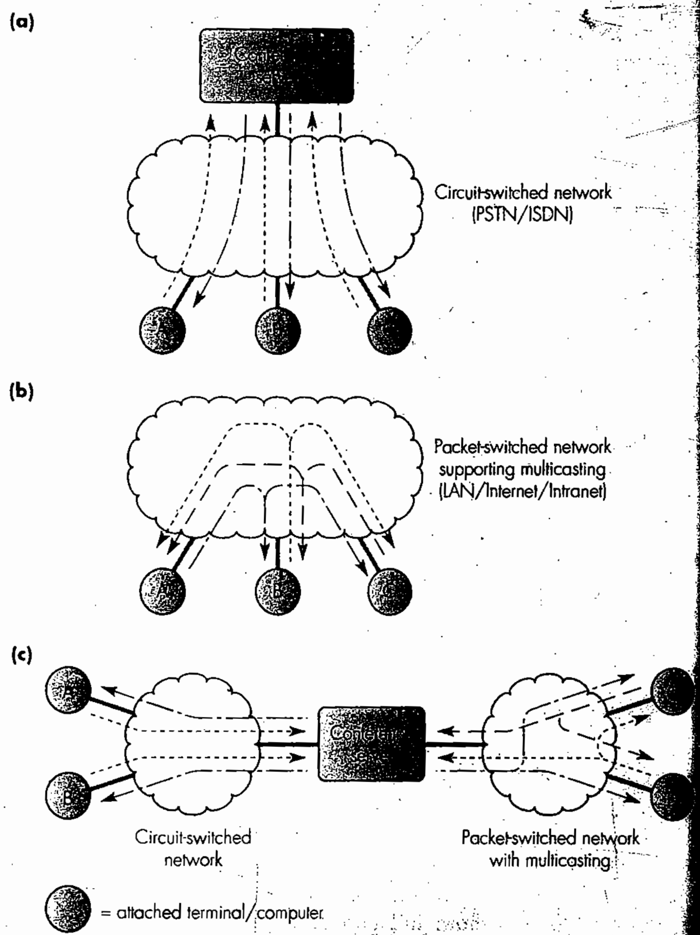


Figure 1.21 Multipoint conferencing modes of operation: (a) centralized; (b) decentralized; (c) hybrid.

- audioconferencing: this involves audio (speech) only;
- videoconferencing: this involves speech and video synchronized and integrated together;
- multimedia conferencing: this involves speech, video, and data integrated together.

With data conferencing, the information flow between the various parties is relatively infrequent. Normally, therefore, the conference server is a general-purpose computer with the conference function implemented in software. With the other three types of conferencing, however, the information flows demand the use of special purpose units. In the case of audioconferencing, the unit is called an **audio bridge** and typical units support from six through to 48 conference participants. With video and multimedia conferencing, the unit is called a multipoint control unit (MCU) and, because of the volume and rate of the information being exchanged, normally, the centralized mode of working is used with both network types.

An MCU consists of two parts: the first is known as the **multipoint controller (MC)** part and is concerned with the establishment of connections to each of the conference participants and with the negotiation of an agreed set of operational parameters – screen resolution, refresh rate, and so on. The second part is known as the **multipoint processor (MP)** and is concerned with the distribution of the information streams generated during the conference. The latter include such functions as the mixing of the various media streams into an integrated stream, voice-activated switching and continuous presence.

When using an audio bridge, a call is scheduled for a particular date, time, and duration and everyone who is to take part in the call is assigned a user ID and password. At the appropriate time, all participants call in and, after they have been verified to join the conference, they can hear and speak to the other participants. In a similar way, when using an MCU, a call is scheduled as for an audio bridge and, once the conference starts, each participant can hear, see, and share data with the other participants. With an MCU, however, in addition to the participants calling in – known as the **dial-in mode** – in some instances, the MCU calls the participants – the **dial-out mode** – which, in general, provides better security.

In the **voice-activated switching mode**, the face of the participant is displayed in a window on the screen of the participant's terminal/computer and, in a second window, is the face of the (remote) participant who is currently talking. When another participant starts to talk, the face of the new speaker replaces the face of the current remote participant. In the event of two (or more) participants starting to talk at the same time, the MCU normally selects the person who speaks the loudest. In the **continuous-presence mode**, however, the remote window is divided into a number of smaller windows, each of which displays the face of the last set of participants who spoke or who are currently speaking. With both modes the speech from all participants is normally mixed into a single stream and hence each participant can always hear what is said by all the other participants.

1.5.5 Network QoS

The operational parameters associated with a communications channel through a network are known as the **network Quality of Service (QoS) parameters** and collectively they determine the suitability of the channel in relation to its use for a particular application. In practice, the QoS parameters associated with a circuit-switched network are different from those associated with a packet-switched network and hence we shall discuss each separately.

Circuit-switched network

The QoS parameters associated with a constant bit rate channel that is set up through a circuit-switched network include:

- the bit rate,
- the mean bit error rate,
- the transmission delay.

The mean **bit error rate** (**BER**) of a channel is the probability of a bit being corrupted during its transmission across the channel in a defined time interval. Hence, for a constant bit rate channel, this equates to the probability of a bit being corrupted in a defined number of bits. A mean BER of 10^{-3}, therefore, means that, on average, for every 1000 bits that are transmitted, 1 of these bits will be corrupted. In some applications, providing the occurrence of bit errors is relatively infrequent, their presence is acceptable while in other applications it is imperative that no residual bit errors are present in the received information. For example, if the application involves speech, then an occasional bit error will go unnoticed but in an application involving the transfer of, say, financial information, it is essential that the received information contains no errors. Hence with such applications, prior to transmission the source information is normally divided into blocks the maximum size of which is determined by the mean BER of the communications channel.

For example, if the mean BER is 10^{-3}, then the number of bits in a block must be considerably less than 1000 otherwise, on average, every block will contain an error and will be discarded. Normally, however, bit errors occur randomly and hence, even with a block size of, say, 100 bits, blocks may still contain an error but the probability of this occurring is considerably less. In general, if the BER probability is P and the number of bits in a block is N then, assuming random errors, the probability of a block containing a bi error, P_B, is given by:

$$P_B = 1-(1-P)^N$$

which approximates to $N \times P$ if $N \times P$ is less than 1.

In practice, most networks – both circuit-switched and packet-switched – provide an **unreliable service** which is also known as a **best-try** or **best-effort** service. This means that any blocks containing bit errors will be discarded either within the network – packet-switched networks – or in the network interface at the destination – both packet-switched and circuit-switched networks. Hence if the application dictates that only error-free blocks are

ample 1.1

Derive the maximum block size that should be used over a channel which has a mean BER probability of 10^{-4} if the probability of a block containing an error – and hence being discarded – is to be 10^{-1}.

Answer:

$$P_B = 1-(1-P)^N$$

Hence $0.1 = 1-(1-10^{-4})^N$ and $N = 950$ bits

Alternatively, $P_B = N \times P$

Hence $0.1 = N \times 10^{-4}$ and $N = 1000$ bits

acceptable, it is necessary for the sending terminal/computer to divide
source information into blocks of a defined maximum size and for the dest
tion to detect when a block is missing. When this occurs the destination n
request that the source send another copy of the missing block. The ser
offered is then said to be a **reliable service**. Clearly, this will introduce a d
so the retransmission procedure should be invoked relatively infreque
which dictates a small block size. This, however, leads to high overheads si
each block must contain the additional information that is associated with
retransmission procedure. Normally, therefore, the choice of block size
compromise between the increased delay resulting from a large block si
and hence retransmissions – and the loss of transmission bandwidth resu
from the high overheads of using a smaller block size.

The **transmission delay** associated with a channel is determined not
by the bit rate that is used but also delays that occur in the terminal/
puter network interfaces (known as codec delays), plus the propagation
of the digital signals as they pass from the source to the destination acros
network. This is determined by the physical separation of the two comn
cating devices and the velocity of propagation of a signal across
transmission medium. In free space, for example, the latter is equal t
speed of light (3×10^8 ms^{-1}) while it is a fraction of this in physical me
typical value being 2×10^8 ms^{-1}.

Notice that the propagation delay in each case is independent of t
rate of the communications channel and, assuming the codec delay re
constant, is the same whether the bit rate is 1 kbps, 1 Mbps, or 1 Gbps.

Packet-switched network

The QoS parameters associated with a packet-switched network include

- the maximum packet size,
- the mean packet transfer rate,

Example 1.2

Determine the propagation delay associated with the [illegible] nication channels:

(i) a connection through a private [illegible]

(ii) a connection through a PSTN of 200 km

(iii) a connection over a satellite channel of 50000 km

Assume that the velocity of propagation of a signal in the case of (i) and (ii) is $2\times10^8\,\text{ms}^{-1}$ and in the case of (iii) $3\times10^8\,\text{ms}^{-1}$.

Answer

Propagation delay T_p = physical separation/velocity of propagation

(i) $T_p = \frac{10^3}{2\times10^8} = 5\times10^{-6}\,\text{s}$

(ii) $T_p = \frac{200\times10^3}{2\times10^8} = 10^{-3}\,\text{s}$

(iii) $T_p = \frac{5\times10^7}{3\times10^8} = 1.67\times10^{-1}\,\text{s}$

- the mean packet error rate,
- the mean packet transfer delay,
- the worst-case jitter,
- the transmission delay.

In a packet-switched network, although the rate at which packets ar transferred across the network is influenced strongly by the bit rate of th interconnecting links, because of the variable store-and-forward delays i each PSE/router, the actual rate of transfer of packets across the network i also variable. Hence the **mean packet transfer rate** is a measure of the averag number of packets that are transferred across the network per second and coupled with the packet size being used, determines the equivalent mean bi rate of the channel.

The **mean packet error rate** or **PER** is the probability of a received packet containing one or more bit errors. It is the same, therefore, as the block error rate associated with a circuit-switched network which we derived in the previous section. Hence it is related to both the maximum packet size and the worst-case BER of the transmission links that interconnect the PSEs/routers that make up the network.

We defined the meaning of the term "mean packet transfer delay" in Section 1.5.3 when we described the operation of packet-mode networks. It is the summation of the mean store-and-forward delay that a packet experiences in each PSE/router that it encounters along a route and the term "jitter" is the worst-case variation in this delay. As we just explained, the transmission delay is the same whether the network operates in a packet mode or a circuit mode and includes the codec delay in each of the two communicating computers and the signal propagation delay.

1.5.6 Application QoS

The network QoS parameters define what the particular network being used provides rather than what the application requires. The application itself, however, also has QoS parameters associated with it. In an application involving images, for example, the parameters may include a minimum image resolution and size, while in an application involving video, the digitization format and refresh rate may be defined. The application QoS parameters that relate to the network include:

- the required bit rate or mean packet transfer rate,
- the maximum startup delay,
- the maximum end-to-end delay,
- the maximum delay variation/jitter,
- the maximum round-trip delay.

For applications involving the transfer of a constant bit rate stream, the important parameters are the required bit rate/mean packet transfer rate, the end-to-end delay, and, equally important, the delay variation/jitter since this can cause problems in the destination decoder if the rate of arrival of the bitstream is variable. For interactive applications, however, the **startup delay** defines the amount of time that elapses between an application making a request to start a session and the confirmation being received from the application at the destination – a server, for example – that it is prepared to accept the request. Hence this includes, in addition to the time required to establish a network connection – if this is required – the delay introduced in both the source and the destination computers while negotiating that the session can take place. As we saw earlier in Section 1.5.1, the round-trip delay is important since, for human–computer interaction to be successful, the delay between a request for some information being made and the start of the information being received/displayed should be as short as possible and, ideally, should be less than a few seconds.

As we can see from the above, for applications that involve the transfer of a constant bit rate stream, a circuit-switched network would appear to be most appropriate since, firstly, the call setup delay is often not important and secondly, the channel provides a constant bit rate service of a known rate.

Conversely, for interactive applications, a connectionless packet-switched network would appear to be most appropriate since with this there is no network call setup delay and any variations in the packet transfer delay are not important.

An example application that illustrates the benefits of a packet-switched network over a circuit-switched network is the transfer of a large file of data from a server computer connected to the Internet to a client PC/workstation in a home. As we showed earlier in Figures 1.2 and 1.18, access to the Internet from home can be by means of a PSTN (with a modem), an ISDN connection, or a cable modem. In the case of a PSTN and an ISDN, these operate in a circuit-switched mode and provide a constant bit rate channel of in the order of 28.8 kbps (PSTN with modem) and 64/128 kbps (ISDN). In contrast, cable modems operate in a packet mode and, as we shall see later in Section 11.2.1, the modems in each of the homes in a cable region time-share the use of a single high bit rate channel/circuit. A typical bit rate of the shared channel is 27 Mbps and the number of concurrent users of the channel may be several hundred. Hence, assuming 270 concurrent users, each user would get a mean data rate of 100 kbps.

With this type of application, however, the main parameter of interest is not the mean data/bit rate but the time to transmit the complete file. With a PSTN and an ISDN, this is directly related to the channel bit rate and the size of the file. With a cable modem, however, although they time-share the use of the 27 Mbps channel, when they gain access to it, the file transfer takes place at the full rate. Hence assuming the file size is 100 Mbits, the minimum time to transmit the file using the different Internet access modes is:

PSTN and 28.8 kbps modem:	57.8 minutes
ISDN at 64 kbps:	26 minutes
ISDN at 128 kbps:	13 minutes
cable modem at 27 Mbps:	3.7 seconds

In the case of a cable modem, if other transfer requests occur during the time the file is being transmitted, then the completion time of each transfer request will increase as they share the use of the channel. Nevertheless, with this type of application, the probability of multiple users requesting a transfer in this short window of time is relatively low.

In many instances, however, this does not mean that the alternative network types cannot be used. For interactive applications, for example, the call setup delay with an ISDN or an ATM network – and a PSTN for local calls – is very fast and, for many applications, quite acceptable. Similarly, for constant bit rate applications, providing the equivalent mean bit rate provided by the network is greater than the input bit rate and the maximum jitter is less than a defined value, then a packet-switched network can be used. To overcome the effect of jitter a technique known as **buffering** is used, the general principles of which are shown in Figure 1.22.

As we show in the figure, the effect of jitter is overcome by retaining a defined number of packets in a memory buffer at the destination before playout of the information bitstream is started. The memory buffer operates using a first-in, first-out (FIFO) discipline and the number of packets retained in the buffer before output starts is determined by the worst-case jitter and the bit rate of the information stream. However, as we show in part (a) of the figure, when using a packet-switched network for this type of application, an additional delay is incurred at the source as the information bitstream is converted into packets. This is known as the **packetization delay** and adds to the transmission delay of the channel. Hence in order to minimize the overall input-to-output delay, the packet size used for an application is made as small as possible but of sufficient size to overcome the effect of the worst-case jitter.

In order to simplify the process of determining whether a particular network can meet the QoS requirements of an application, a number of standard application **service classes** have been defined. Associated with each service class is a

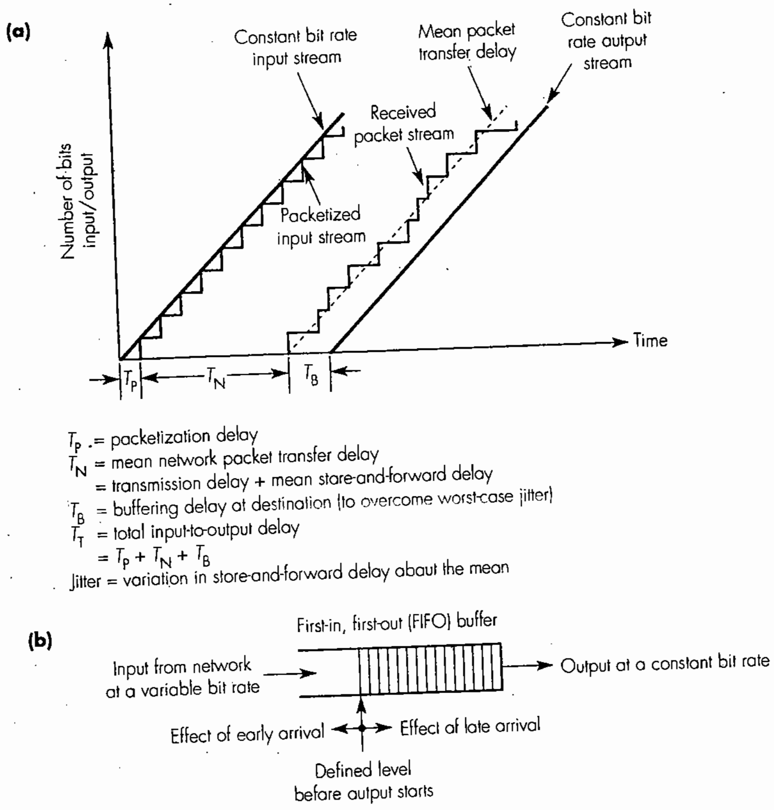

(a)
Constant bit rate input stream
Mean packet transfer delay
Constant bit rate output stream
Received packet stream
Packetized input stream
Number of bits input/output
Time
T_P
T_N
T_B
T_P = packetization delay
T_N = mean network packet transfer delay
= transmission delay + mean store-and-forward delay
T_B = buffering delay at destination (to overcome worst-case jitter)
T_T = total input-to-output delay
$= T_P + T_N + T_B$
Jitter = variation in store-and-forward delay about the mean
(b)
First-in, first-out (FIFO) buffer
Input from network at a variable bit rate
Output at a constant bit rate
Effect of early arrival
Effect of late arrival
Defined level before output starts

Figure 1.22 Transmission of a constant bit rate stream over a packet-switched network: (a) timing schematic; (b) FIFO buffer operation.

mple 1.3

A packet-switched network with a worst-case jitter of 10 ms is to be used for a number of applications each of which involve a constant bit rate information stream. Determine the minimum amount of memory that is required at the destination and a suitable packet size for each of the following input bit rates. It can be assumed that the mean packet transfer rate of the network exceeds the equivalent input bit rate in each case.

(i) 64 kbps

(ii) 256 kbps

(iii) 1.5 Mbps.

Answer:

(i) At 64 kbps, 10 ms = 640 bits
Hence choose a packet size of, say, 800 bits with a FIFO buffer of 1600 bits – 2 packets – and start playout of the bitstream after the first packet has been received.

(ii) At 256 kbps, 10 ms = 2560 bits
Hence choose a packet size of, say, 2800 bits with a FIFO buffer of 4800 bits.

(iii) At 1.5 Mbps, 10 ms = 15000 bits
Hence choose a packet size of, say, 16000 bits with a FIFO buffer of 32 000 bits.

Notice that if the computed packet size exceeds the network maximum packet size, then the equivalent number of packets must be sent before playout starts. For example, if the maximum network packet size was 8000 bits, then for case (iii) above playout would not start until two packets have been received and the FIFO buffer should hold four packets.

specific set of QoS parameters and a network can either meet this set of parameters or not. Also, for networks that support a number of different service classes – the Internet for example – in order to ensure the QoS parameters associated with each class are met, the packets relating to each class are given a different priority. It is then possible to treat the packets relating to each class differently.

For example, as we shall see in Chapter 9, in the Internet, packets relating to multimedia applications involving real-time streams are given a higher priority than the packets relating to applications such as email. Typically, packets containing real-time streams such as audio and video are also more sensitive to delay and jitter than the packets containing textual information. Hence during periods of network congestion, the packets containing real-time streams are transmitted first. Packets containing video are more sensitive to packet loss than packets containing audio and hence are given a higher priority.

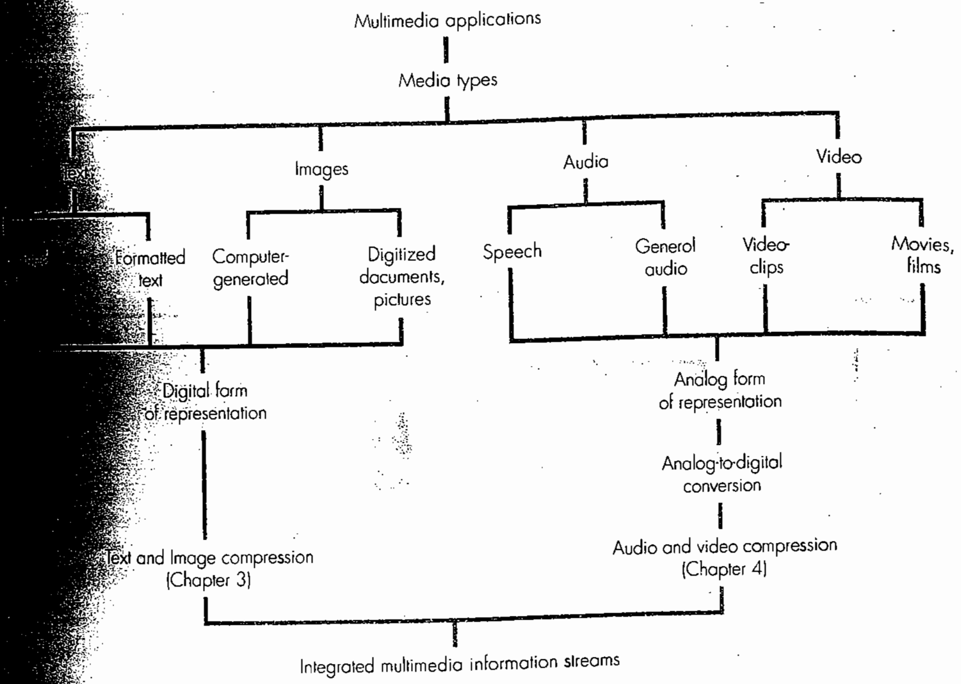
Summary

In this chapter we have discussed:

- the different types of media that are used in multimedia applications,
- the different types of communication networks that are used to support these applications,
- a selection of the different types of application.

Media types

The different types of media that are used in multimedia applications are summarized in Figure 1.23. In some applications only a single type of medium is involved while in others multiple media types are used. As we described in Section 1.2, the basic form of representation of both text and images consists of a block of binary codewords. Each codeword consists of a fixed number of binary digits and represents, for example, a single character (text) or a single picture element (image). In contrast, the basic form of



Multimedia applications
Media types
Text
Images
Audio
Video
Formatted text
Computer-generated
Digitized documents, pictures
Speech
General audio
Video-clips
Movies, films
Digital form of representation
Analog form of representation
Analog-to-digital conversion
Text and Image compression (Chapter 3)
Audio and video compression (Chapter 4)
Integrated multimedia information streams

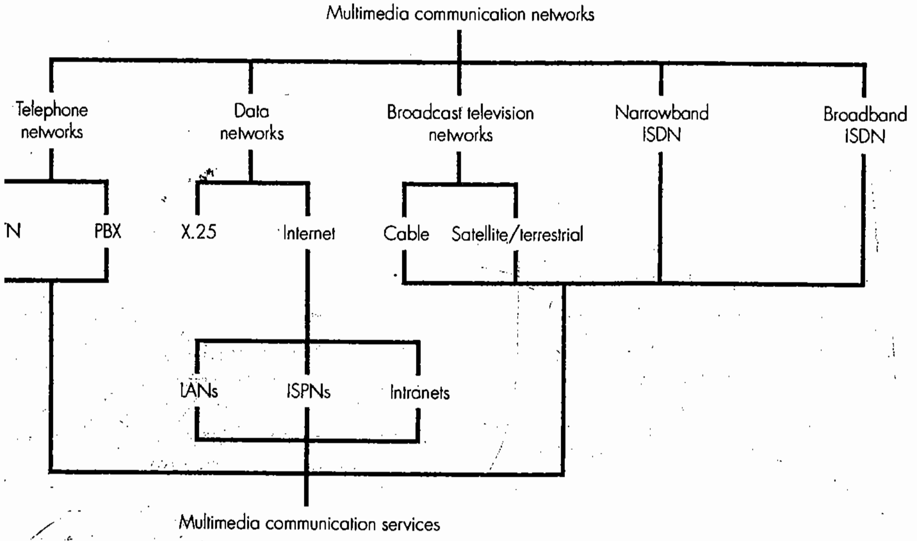
ure 1.23 Alternative types of media used in multimedia applications.

representation of both audio and most video signals is in the form of a analog signal whose amplitude and frequency varies continuously with time. Hence in applications that involve audio and video integrated with tex and/or images, it is necessary first to convert the audio and video signals into a digital form. We shall describe the digital form of representation of all four media types in more detail in the next chapter.

In practice, the bandwidth associated with the basic form of representation of the different media types is greater than the bandwidth of the communication networks that are used to support multimedia applications. Hence in most applications, a technique known as compression is applied to the source information prior to it being transferred over the network. We shall describe a selection of the different compression algorithms that are used with text and images in Chapter 3 and a selection of those used with audio and video in Chapter 4.

Network types

The different types of communication networks that are used to support multimedia applications are summarized in Figure 1.24 and a selection of the applications that each network type supports are shown in Figure 1.25.


Multimedia communication networks
Telephone networks
Data networks
Broadcast television networks
Narrowband ISDN
Broadband ISDN
N
PBX
X.25
Internet
Cable
Satellite/terrestrial
LANs
ISPNs
Intranets
Multimedia communication services

TN = public switched telephone network
X = private branch exchange
DN = integrated services digital network

LANs = local area networks
ISPNs = internet service provider networks

re 1.24 Multimedia communication networks.

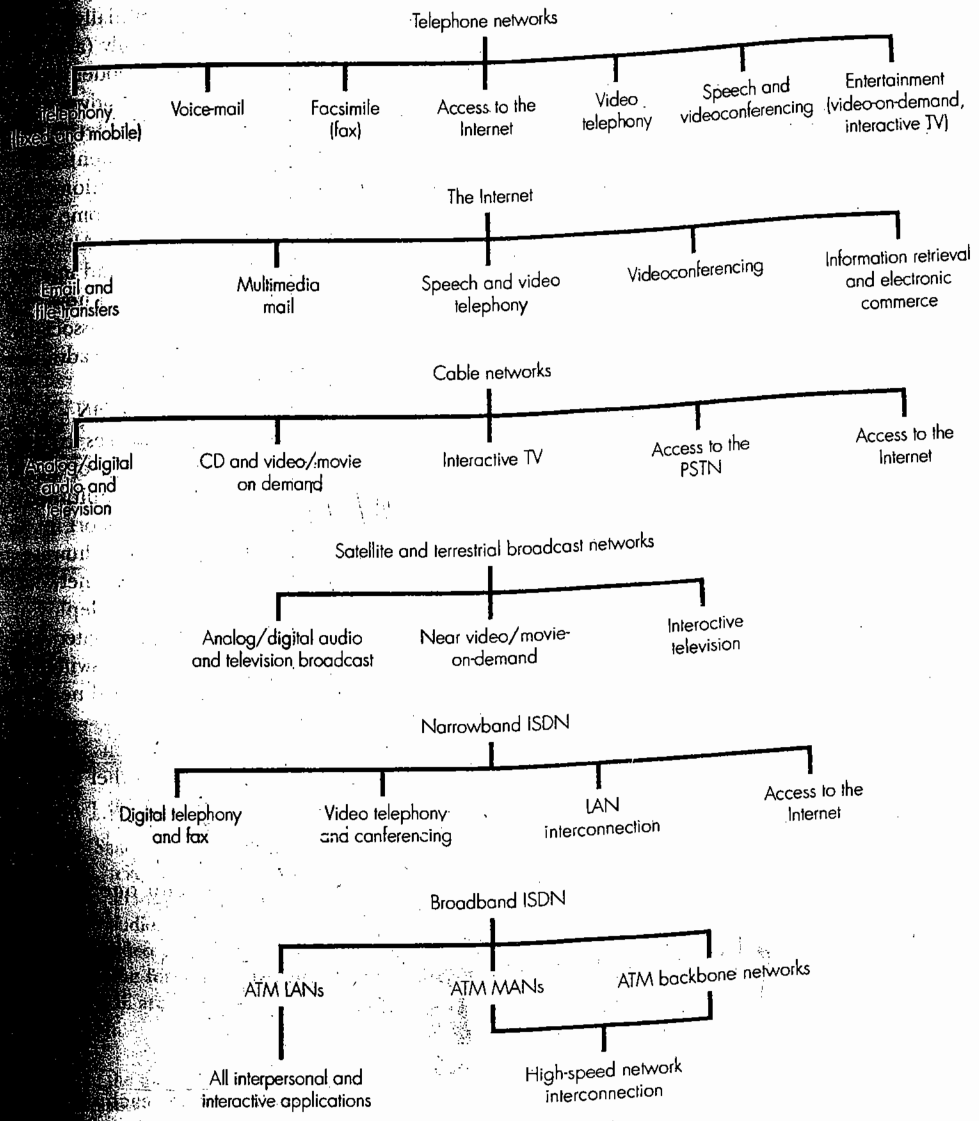

Telephone networks
Telephony (fixed and mobile)
Voice-mail
Facsimile (fax)
Access to the Internet
Video telephony
Speech and videoconferencing
Entertainment (video-on-demand, interactive TV)
The Internet
Email and file transfers
Multimedia mail
Speech and video telephony
Videoconferencing
Information retrieval and electronic commerce
Cable networks
Analog/digital audio and television
CD and video/movie on demand
Interactive TV
Access to the PSTN
Access to the Internet
Satellite and terrestrial broadcast networks
Analog/digital audio and television broadcast
Near video/movie-on-demand
Interactive television
Narrowband ISDN
Digital telephony and fax
Video telephony and conferencing
LAN interconnection
Access to the Internet
Broadband ISDN
ATM LANs
ATM MANs
ATM backbone networks
All interpersonal and interactive applications
High-speed network interconnection

Figure 1.25 Multimedia communication networks and their services.

In the case of telephone networks, these were designed initially to provide a switched telephone service both nationally/internationally (a PSTN) and at a single site (a PBX). Also, with the advent of the first modems, the provided a low bit rate switched data service. Later, as a result of advances i modem technology and related compression algorithms, the transfer of digitized documents (facsimile) was supported. These developments have continued and the PSTN now supports a wide range of applications that involve all four media types either singly or integrated together in some way.

Similarly, with data networks – and in particular the Internet. Although designed initially to support applications that involve only text and images, the advances in compression algorithms means that the Internet now supports a similar range of multimedia applications. The applications associated with broadcast television networks have been enhanced by the same advances in modem technology and compression algorithms.

The last two network types – narrowband and broadband ISDN – were both designed from the outset to provide an enhanced set of services to those provided by a PSTN. A narrowband ISDN, for example, has a range of digital access alternatives which, collectively, support a wide range of multimedia applications. Broadband ISDN was designed from the start to support multimedia applications that involve all four media types. In practice, however, because of the wider set of (multimedia) applications that the other networks can now support, broadband ISDN is not as yet widely deployed. Nevertheless, the basic transmission and switching technology associated with it – known as the asynchronous transfer mode (ATM) and cell switching respectively – has been used in high-speed LANs and for high-speed network interconnection.

We shall discuss in some detail the operation of these various network types in Chapters 7–11 and the communication protocols and other application-related issues that are associated with them in Chapters 12–14.

Multimedia applications

We have chosen to place all multimedia applications into one of three categories:

- interpersonal communications,
- interactive applications over the Internet,
- entertainment applications.

There are a number of applications in each category each of which involves various media types. A selection of the applications, together with the type of media involved, are summarized in Table 1.1. Note that each type of application can be supported by a number of different network types. For example, video telephony is supported by a PSTN, the Internet, and both narrowband and broadband ISDN.

Table 1.1 Multimedia applications summary.

Category	*Media*	*Application descriptions*
Interpersonal communications	Speech	Telephony, voice-mail, teleconferencing
	Image	Facsimile
	Text	Electronic mail
	Text and images	Computer-supported cooperative working (CSCW)
	Speech and video	Video telephony, video mail, videoconferencing
	Text, image, audio and video	Multimedia electronic mail, multiparty video games etc.
Interactive applications over the Internet	Text, image, audio and video	Information retrieval (news, weather, books, magazines, video games, product literature etc.) Electronic commerce
Entertainment services	Audio and video	Audio/CD-on-demand Movie/video-on-demand Near movie/video-on-demand Analog and digital television broadcasts Interactive television

Exercises

Section 1.2

State the basic form of representation of:

(i) text,
(ii) an image,
(iii) audio,
(iv) video.

State the form of representation that is used when all are integrated together and give your reason.

State the meaning of the term "bits per second" in relation to digitized audio and video. What is the meaning of the term "compression" and why is compression used?

Section 1.3

1.3 With the aid of a diagram, explain the meaning of the following terms relating to a switched telephone network:
  (i) POTS,
  (ii) local exchange/end office,
  (iii) PBX,
  (iv) mobile switching center,
  (v) international gateway exchange.

1.4 Explain why a pair of modems is required to transmit a digital signal over a PSTN. With the aid of a diagram, show the location of the two modems when two digital devices communicate over a PSTN and the types of signal – analog or digital – that are used over each part of the access circuit.

Show in the form of a diagram how a high-speed modem provides a range of additional services to basic telephony and data services. State a typical access bit rate that is used.

With the aid of a diagram, describe how the following gain access to the Internet:
(i) a user at home or in a small business,
(ii) a distributed community of users around a single site or campus,
(iii) a distributed set of users all attatched to the same enterprise-wide private network. Why is the latter sometimes called an intranet?

Explain why most data networks operate in a packet mode. Hence explain why services involving audio and video are supported.

State the aim of all broadcast television networks.

With the aid of diagrams, explain how additional services are provided with
(i) a cable distribution network and
(ii) a satellite/terrestrial broadcast network.

With aid of a diagram, explain the meaning of the following terms relating to ISDN:
(i) digital subscriber line,
(ii) basic rate access,
(iii) aggregation,
(iv) primary rate access,
(v) p × 64.

0 Explain the meaning of the term "broadband" in relation to a B-ISDN and why the deployment of such networks has been delayed.

11 Explain the meaning of the following terms relating to a B-ISDN:
(i) a cell,
(ii) ATM,
(iii) cell switching.

ection 1.4

12 In relation to speech-only interpersonal communications involving both public (PSTN/ISDN) and private (PBX) networks, with the aid of a diagram explain how voice-mail and teleconferencing are supported. Include in your descriptions the role of a voice-mail server and an audio bridge.

1.13 With the aid of a diagram, explain the function of a telephony gateway in relation to Internet telephony. Hence state the origin of the term "voice over IP" (VoIP).

1.14 Describe the principal operation of a fax machine and why modems are required. What is the meaning of the term "PC fax"?

1.15 Show in the form of a diagram the networks and essential items of equipment that are used to send an email message from a PC user at home to
(i) a PC attached to a site/campus LAN,
(ii) a PC attached to a site/campus LAN,
(iii) a PC attached to an enterprise-wide private network/intranet.

1.16 With the aid of a diagram, explain the basic principles behind computer-supported cooperative working (CSCW). Include in your explanation the role of the following:
(i) shared whiteboard,
(ii) whiteboard program,
(iii) change-notification,
(iv) update control.

1.17 Use a schematic diagram to illustrate each of the following:
(i) a two-party video phone call,
(ii) a videoconferencing call using an MCU,
(iii) a videoconferencing call using a broadcast network. Show on your diagrams the bandwidth requirements of the access circuits in each case.

1.18 Explain the role of an MCU in relation to a videoconferencing session involving multiple geographically distributed videoconferencing studios.
Quantify the bandwidth implications of locating the MCU at one of the sites.

1.19 With multimedia electronic mail, when a message in a number of different formats is sent – for example, speech-with-video, speech-only and text – normally the first version in output first. Why is this?

1.20 Use a schematic diagram to explain the meaning of the following terms relating to interactive applications over the inernet:
(i) a Web server,
(ii) a browser,
(iii) the World Wide Web.

1.21 In relation to a document stored on a Web server, explain the meaning of the following terms:
(i) Web page,
(ii) home page,
(iii) hyperlink,
(iv) URL,
(v) HTML.

1.22 With the aid of a diagram, explain the principle of operation of movie/video-on-demand. Identify the bandwidth requirements associated with this type of application.

1.23 What is the limiting factor associated with movie/video-on-demand? Explain how this effect is reduced with near movie-on-demand.

1.24 Explain the meaning of the term "interactive television" and, with the aid of diagrams, how this is using either a cable distribution network or a satellite/terrestrial broadcast network. Explain how Internet access is obtained with each network type.

Section 1.5

1.25 What is the meaning of the following terms relating to the different types of media used in multimedia communications:
(i) block-mode media,
(ii) continuous media,
(iii) streaming,
(iv) constant bit rate,
(v) variable bit rate?

1.26 With the aid of diagrams, explain the meaning of the following operational modes of a communication channel:
(i) simplex,
(ii) half-duplex,
(iii) duplex,
(iv) broadcast
(v) multicast,
(vi) asymmetric and symmetric.

1.27 With the aid of a diagram explain the principle of operation of a circuit-mode network. Include in your explanation the need for signaling (messages) and the overheads of the connection setup delay associated with this.

1.28 With the aid of diagrams, explain the principle of operation of a CO packet-mode network. Include in your explanation the need for a virtual connection/circuit, a virtual circuit identifier, and a routing table.

1.29 In relation to a CL packet-mode network, explain the meaning of the following terms:
(i) best effort service,
(ii) store-and-forward display,
(iii) mean packet transfer delay,
(iv) jitter.

1.30 With the aid of diagrams, explain the following operational modes of multipoint conferencing:
(i) centralized,
(ii) decentralized,
(iii) hybrid.

1.31 Identify the application domain and explain the operation of the following types of conference server:
(i) general purpose,
(ii) audio bridge,
(iii) multipoint control unit,

1.32 In relation to conference servers, explain the meaning of the following terms:
(i) multipoint controller,
(ii) multipoint processor,
(iii) dial-in mode,

   (iv) dial-out mode,
   (v) voice activated switching,
   (vi) continuous presence.

1.33 Identify and explain the meaning of the key QoS parameters associated with the following network types:
   (i) circuit-switched,
   (ii) packet-switched.

1.34 Define the BER probability of a transmission line/channel. How does this influence the maximum block size to be used with the line/channel?

1.35 Define the transmission delay of a transmission line/channel. First identify the individual sources of delay that contribute to this.

1.36 Identify and explain the meaning of the key parameters associated with application QoS.

1.37 With the aid of a diagram explain the meaning of the following terms relating to packet-switched network:
   (i) packetization delay,
   (ii) mean packet transfer delay,
   (iii) jitter.

   Hence describe how the effects on a constan bit rate stream of packetization delay and jitte can be overcome by buffering.

1.38 A Web page of 10 Mbytes is being retrieve from a Web server. Assuming negligible delay within the server and trunk network, quantify the time to transfer the page over the followin types of access circuit:
   (i) a PSTN modem operating at 28.8 kbps,
   (ii) an aggregated basic rate access line of 128 kbps,
   (iii) a primary rate ISDN access line of 1.5 Mbps,
   (iv) a high-speed modem operating at 6 Mbps,
   (v) a cable modem operating at 27 Mbps.

1.39 Discuss the term "application service classes". Include in your dicussion how packets belonging to different classes are treated within the network.

2

Multimedia information representation

2.1 Introduction

All types of multimedia information are stored and processed within a computer in a digital form. In the case of textual information consisting of strings of characters entered at a keyboard, for example, each character is represented by a unique combination of a fixed number of bits – known as a **codeword** – and hence the complete text by a string of such codewords. Similarly, computer-generated graphical images are made up of a mix of lines, circles, squares, and so on, each represented in a digital form. A line, for example, is represented by means of the start and end coordinates of the line relative to the complete image, each coordinate being defined in the form of a pair of digital values.

In contrast, devices such as microphones and many video cameras produce electrical signals whose amplitude varies continuously with time, the amplitude of the signal at any point in time indicating the magnitude of the sound-wave/image-intensity at that instant. As we indicated in Section 1.2, a signal whose amplitude varies continuously with time is known as an **analog signal**. In order to store and process such signals – and hence types of media – within a computer, it is necessary first to convert any time-varying analog signals into a digital form. In addition, the signals used to operate devices such as loudspeakers – for speech and audio – and computer monitors – for the display of digitized images for example – are also analog signals. Thus digital values representing such media types must be converted back again into a corresponding time-varying analog form on output from the computer.

The conversion of an analog signal into a digital form is carried out using an electrical circuit known as a **signal encoder**. This, as we will expand upon in the next section, operates by first **sampling** the amplitude of the analog signal at repetitive time intervals and then converting the amplitude of each sample into a corresponding digital value. Similarly, the conversion of the stored digitized samples relating to a particular media type into their corresponding time-varying analog form is performed by an electrical circuit known as a **signal decoder**.

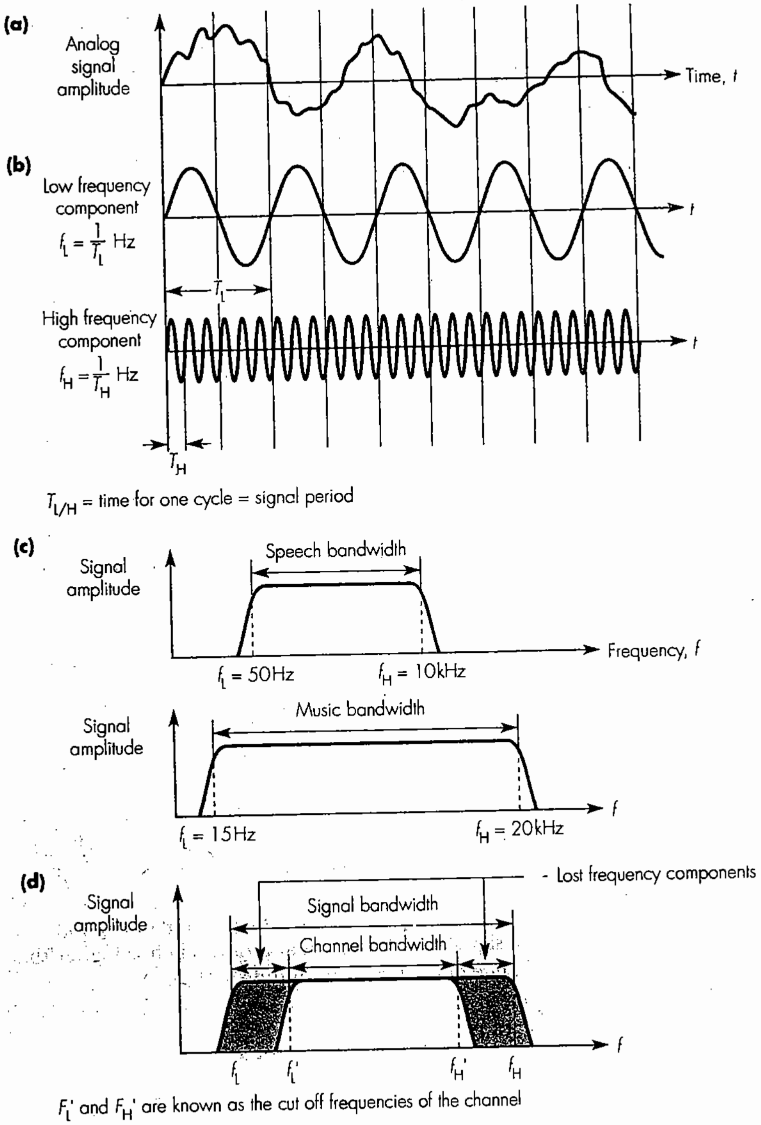
In this way, all the media types associated with the various multimedia applications discussed in the last chapter are stored and processed within a computer in an all-digital form. This means that the different media types can be readily integrated together and, equally important, the resulting integrated bitstream can be transmitted over a single all-digital communications network. This chapter is concerned with the way the different media types are represented in their digital form and, where appropriate, the conversion operations that are used. Before discussing this, however, it will be helpful if we first gain an understanding of the general principles relating to how analog signals are converted into their digital form and vice versa.

2.2 Digitization principles

2.2.1 Analog signals

The general properties relating to any time-varying analog signal are shown in Figure 2.1. As we can see in part (a) of the figure, the amplitude of such signals varies continuously with time. In addition, a mathematical technique known as **Fourier analysis** can be used to show that any time-varying analog signal is made up of a possibly infinite number of single-frequency sinusoidal signals whose amplitude and phase vary continuously with time relative to each other. As an example, the highest and lowest frequency components of the signal shown in Figure 2.1(a) may be those shown in Figure 2.1(b).

The range of frequencies of the sinusoidal components that make up a signal is called the **signal bandwidth** and two examples are shown in part (c) of the figure. These relate to an audio signal, the first a speech signal and the second a music signal produced by, say, an orchestra. In terms of speech, humans produce sounds – which are converted into electrical signals by a



(a)
Analog signal amplitude
Time, t
(b)
Low frequency component
$f_L = \frac{1}{T_L}$ Hz
T_L
High frequency component
$f_H = \frac{1}{T_H}$ Hz
T_H
t
$T_{L/H}$ = time for one cycle = signal period
(c)
Signal amplitude
Speech bandwidth
Frequency, f
f_L = 50Hz
f_H = 10kHz
Music bandwidth
f_L = 15Hz
f_H = 20kHz
(d)
Lost frequency components
Signal bandwidth
Channel bandwidth
f_L
f_L'
f_H'
f_H
f_L' and f_H' are known as the cut off frequencies of the channel

Figure 2.1 Signal properties: (a) time-varying analog signal; (b) sinusoidial frequency components; (c) signal bandwidth examples; (d) effect of a limited bandwidth transmission channel.

microphone – that are made up of a range of sinusoidal signals varying quency between 50 Hz and 10 kHz. In the case of a music signal, howeve range of signals is wider and varies between 15 Hz and 20 kHz, this being parable with the limits of the sensitivity of the ear.

Ideally, when an analog signal is being transmitted through a ne the bandwidth of the transmission channel – that is, the range of freque the channel will pass – should be equal to or greater than the bandwid the signal. If the bandwidth of the channel is less than this, then some o low and/or high frequency components will be lost thereby degradin quality of the received signal. This type of transmission channel is call **bandlimiting channel** and its effect is shown in Figure 2.1 (d).

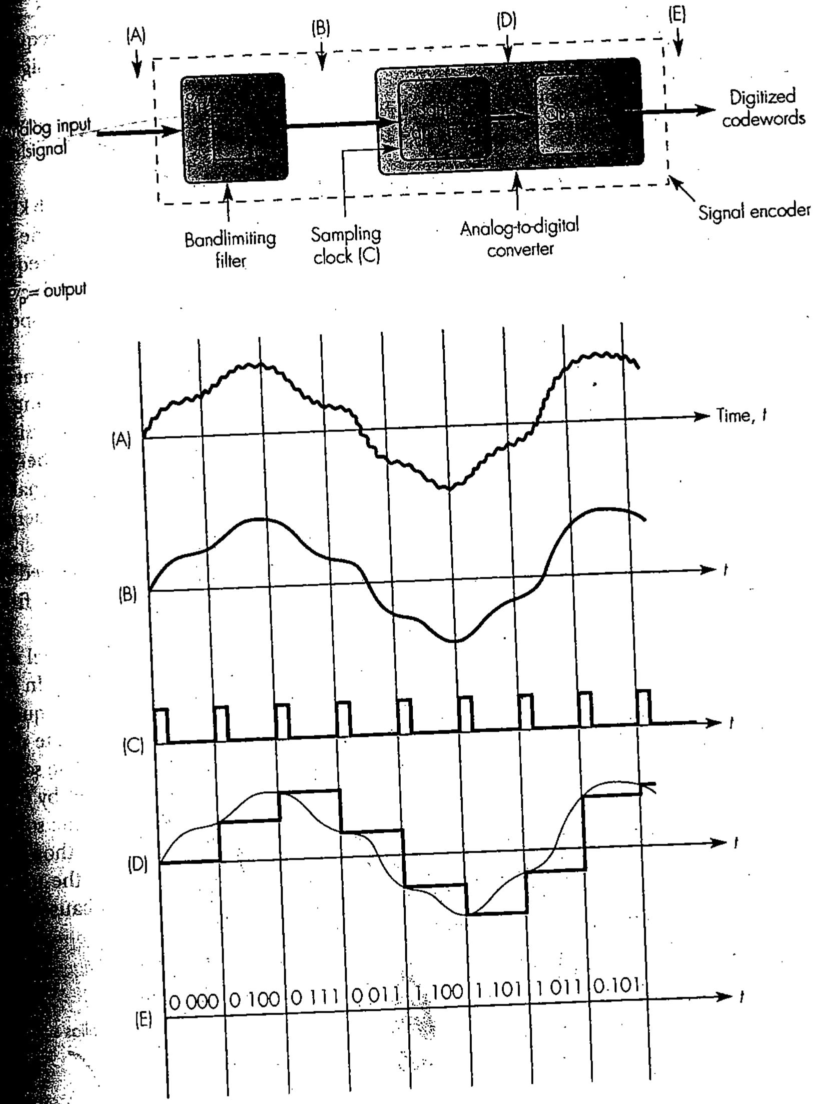
2.2.2 Encoder design

As indicated in Section 2.1, the conversion of a time-varying analog sign of which an audio signal is an example – into a digital form is carried using an electronic circuit known as a (signal) encoder. The principles o encoder are shown in Figure 2.2 and, as we can see in part (a), it consist two main circuits: a **bandlimiting filter** and an **analog-to-digital conver** **(ADC)**, the latter comprising a sample-and-hold and a quantizer. A typi **waveform set** for a signal encoder is shown in part (b) of the figure. The r of the bandlimiting filter, as we shall see below, is to remove selected high frequency components from the source signal (A). The output of the fil (B) is then fed to the **sample-and-hold** circuit which, as its name implies used to sample the amplitude of the filtered signal at regular time interv (C) and to hold the sample amplitude constant between samples (D). Th in turn, is fed to the **quantizer** circuit which converts each sample amplitu into a binary value known as a codeword (E).

The most significant bit of each codeword indicates the polarity (sign) the sample, positive or negative relative to the zero level. Normally, a binar indicates a positive value and a binary 1 a negative value. Also, as we c deduce from the time-related set of waveforms shown in Figure 2.2(b), to re resent the amplitude of a time-varying analog signal precisely, requires firs the signal to be sampled at a rate which is higher than the maximum rate change of the signal amplitude and secondly, the number of different qua zation levels used to be as large as possible. We shall consider each of the requirements separately.

Sampling rate

In relation to the sampling rate, the **Nyquist sampling theorem** states that: i order to obtain an accurate representation of a time-varying analog signal, i amplitude must be sampled at a minimum rate that is equal to or greate than twice the highest sinusoidal frequency component that is present in th signal. This is known as the **Nyquist rate** and is normally represented as eithe Hz or, more correctly, **samples per second** (sps). Sampling a signal at a rate



2.2 Signal encoder design: (a) circuit components; (b) associated waveform set.

which is lower than the Nyquist rate results in additional frequency components being generated that are not present in the original signal which, i turn, cause the original signal to become **distorted**.

The distortion caused by sampling a signal at a rate lower than th Nyquist rate is best illustrated by considering the effect of undersampling a single-frequency sinusoidal signal as shown in Figure 2.3.

In this example, the original signal is assumed to be a 6 kHz sinewave which is sampled at a rate of 8 ksps. Clearly this is lower than the Nyquist rate of 12 ksps (2 × 6 kHz) and, as we can see, results in a lower-frequency 2 kHz signal being created in place of the original 6 kHz signal. Because of this such signals are called **alias signals** since they replace the corresponding original signals.

In general, this means that all frequency components present in the original signal that are higher in frequency than half the sampling frequency being used (in Hz), will generate related lower-frequency alias signals which will simply add to those making up the original source signal thereby causing it to become distorted. However, by first passing the source signal through a bandlimiting filter which is designed to pass only those frequency components up to that determined by the Nyquist rate, any higher-frequency components in the signal which are higher than this are removed before the signal is sampled. Because of this function the bandlimiting filter is also known as an **antialiasing filter**.

In practice, as we indicated earlier, the transmission channel used/available often has a lower bandwidth than that of the source signal. In such cases, in order to avoid distortion, it is the bandwidth – and hence frequency range – of the transmission channel that determines the sampling rate used rathe than the bandwidth of the source signal. Since in such cases the source signa may have higher frequency components than those dictated by the Nyquis rate of the transmission channel, it is necessary first to pass the source signa through a bandlimiting filter which is designed to pass only those sinusoida frequency components which are within the bandwidth of the transmissio channel. In this way, the generation of any alias signals caused by **undersampling** the source signal is avoided.

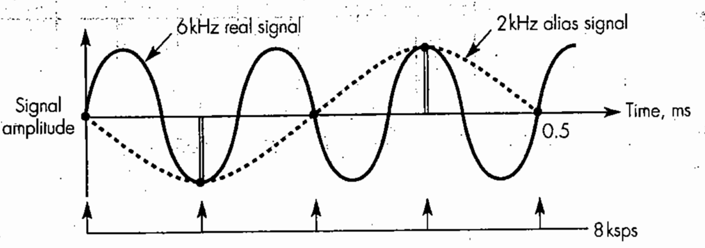


Figure 2.3 Alias signal generation due to undersampling.

Example 2.1

[illegible] the rate of the sampler and the bandwidth of the bandlimit[illegible] an encoder which is to be used for the digitization of an [illegible] signal which has a bandwidth from 15 Hz through to 10 kHz [illegible] digitized signal:

[illegible] is to be stored within the memory of a computer,

[illegible] is to be transmitted over a channel which has a bandwidth from 200 Hz through to 3.4 kHz.

The Nyquist sampling rate must be at least twice the highest frequency [illegible] of the signal or transmission channel. Hence:

[illegible] The sampling rate must be at least 2 × 10 kHz = 20 kHz or 20 ksps and the bandwidth of the bandlimiting filter is from 0 Hz through to 10 kHz.

[illegible] The sampling rate must be at least 2 × 3.4 kHz = 6.8 kHz or 6.8 ksps and the bandwidth of the bandlimiting filter is from 0 Hz through to 3.4 kHz.

In practice, it should be noted that, because of imperfections in filters, some higher-frequency components above the filter cut-off frequency [illegible] passed and hence the sampling rate is normally higher than the [illegible] values. In the case of (ii), for example, it is common to [illegible] frequency components of up to 4 kHz may be passed by the [illegible] filter and hence a sampling rate of 8 ksps is normally used.

Quantization intervals

To represent in a digital form the amplitudes of the set of analog samples shown earlier in Figure 2.2 precisely, would require an infinite number of binary digits since, when a finite number of digits is used, each sample can only be represented by a corresponding number of discrete levels. The effect of using a finite number of bits can be seen by considering the example shown in Figure 2.4. In this example we use just three bits to represent each sample including a sign bit. This results in four positive and four negative quantization intervals, the two magnitude bits being determined by the particular quantization interval the analog input signal is in at the time of each sample.

As we can deduce from part (a) of the figure, if $V_{\max}$ is the maximum positive and negative signal amplitude and n is the number of binary bits used, then the magnitude of each **quantization interval**, q, is given by

$$q = \frac{2V_{\max}}{2^n}$$

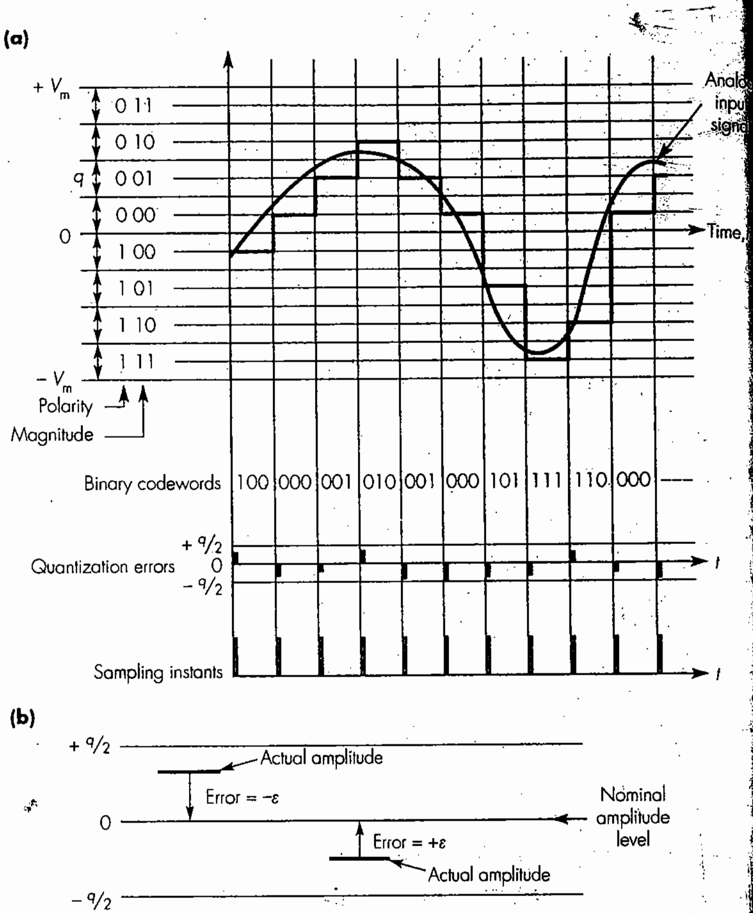


Figure 2.4 Quantization procedure: (a) source of errors; (b) noise polarit

Also, as we can see, a signal anywhere within a quantization interval wi be represented by the same binary codeword. This means that each cod word corresponds to a nominal amplitude level which is at the center of th corresponding quantization interval. Hence the actual signal level may diff from this by up to plus or minus $q/2$.

The difference between the actual signal amplitude and the correspond ing nominal amplitude is called the **quantization error** and, for the exampl shown in Figure 2.4, the quantization error values are shown expanded in par (b) of the figure. Normally, the error values will vary randomly from sample t sample and hence the quantization error is also known as **quantization noise**

the term "noise" being used in electrical circuits to refer to a signal whose amplitude varies randomly with time.

Another related factor which influences the choice of the number of quantization intervals used for a particular signal is its smallest amplitude relative to its peak amplitude. With high-fidelity music, for example, it is important to be able to hear very quiet passages without any distortion created by quantization noise. The ratio of the peak amplitude of a signal to its minimum amplitude is known as the **dynamic range** of the signal, ***D***. Normally it is quantified using a logarithmic scale known as **decibels** or **dB**:

$$D = 20\log_{10}(V_{\max}/V_{\max})\text{dB}$$

Hence when determining the quantization interval – and thus number of bits to be used – it is necessary to ensure that the level of quantization noise relative to the smallest signal amplitude is acceptable.

Example 2.2

An analog signal has a dynamic range of 40 dB. Determine the magnitude of the quantization noise relative to the minimum signal amplitude if the quantizer uses (i) 6 bits and (ii) 10 bits.

Answer:

$$D = 20\log_{10}\frac{V_{\max}}{V_{\min}}\text{dB} \quad \text{Quantization noise} = \pm\frac{q}{2} = \pm\frac{V_{ma}}{2}$$

$$\text{Hence } 40 = 20\log_{10}\frac{V_{\max}}{V_{\min}}$$

$$\text{and } V_{\min} = \frac{V_{\max}}{100}$$

(i) $n = 6$ bits

$$\text{Hence } \frac{q}{2} = \pm\frac{V_{\max}}{2^6} = \pm\frac{V_{\max}}{64}$$

(ii) $n = 10$ bits

$$\text{Hence } \frac{q}{2} = \pm\frac{V_{\max}}{2^{10}} = \pm\frac{V_{\max}}{1024}$$

As we can see from these values, with 6 bits the quantization noise is greater than $V_{\min}$ and hence is unacceptable. With 10 bits, however, the quantization noise is an order of magnitude less than $V_{\min}$ and hence will have a much reduced effect.

2.2.3 Decoder design

As indicated in Section 2.1, although analog signals are stored, processed and transmitted in a digital form, normally, prior to their output, they must be converted back again into their analog form; loudspeakers, for example, are driven by an analog current signal. The electronic circuit that performs this conversion operation is known as a (signal) decoder, the principles of which are shown in Figure 2.5.

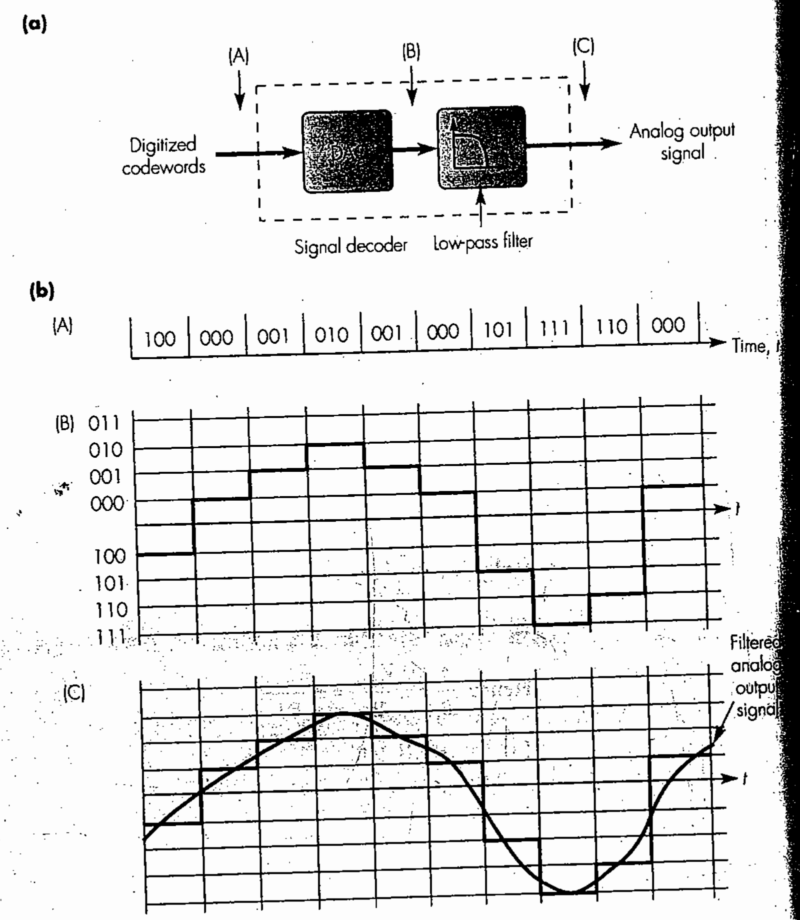


Figure 2.5 Signal decoder design: (a) circuit components; (b) associated waveform set.

First, each digital codeword (A) is converted into an equivalent analog sample using a circuit called a **digital-to-analog converter** or **DAC**. This produces the signal shown in (B), the amplitude of each level being determined by the corresponding codeword. Since this is a time-varying signal, as indicated earlier, Fourier analysis can be used to show that this type of signal comprises not just the sinusoidal frequency components that make up the original (filtered) analog signal, but also an infinite number of additional higher-frequency components. In order to reproduce the original signal, the output of the DAC is passed through a **low-pass filter** which, as its name implies, only passes those frequency components that made up the original filtered signal (C). Normally, the high-frequency cut-off of the low-pass filter is made the same as that used in the bandlimiting filter of the encoder. Because of its function, the low-pass filter is also known as a **recovery** or **reconstruction filter.**

Finally, since in most multimedia applications involving audio and video the communications channel is two-way simultaneous, the terminal equipment must support both input and output simultaneously. Hence the audio/video signal encoders and decoders in each terminal equipment are often combined into a single unit called an **audio/video encoder-decoder** or simply an **audio/video codec.**

2.3 Text

Essentially, there are three types of text that are used to produce pages of documents:

- **unformatted text:** this is also known as **plaintext** and enables pages to be created which comprise strings of fixed-sized characters from a limited character set;
- **formatted text:** this is also known as **richtext** and enables pages and complete documents to be created which comprise of strings of characters of different styles, size, and shape with tables, graphics, and images inserted at appropriate points;
- **hypertext:** this enables an integrated set of documents (each comprising formatted text) to be created which have defined linkages between them.

We shall discuss the three types separately.

2.3.1 Unformatted text

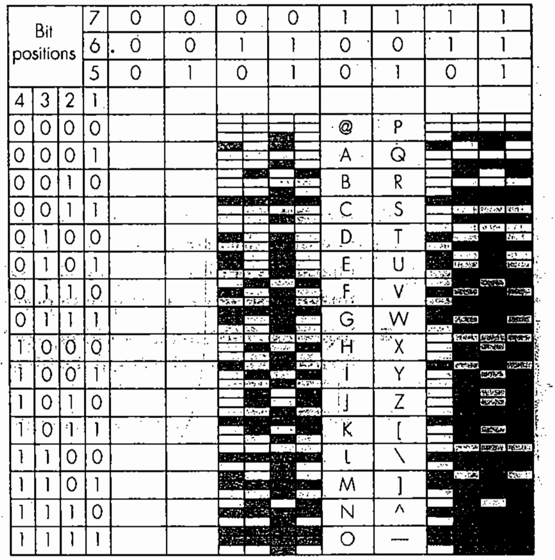
Figure 2.6 illustrates two examples of character sets that are widely used to create pages consisting of unformatted text strings.

The table shown in part (a) tabulates the set of characters that are available in the **ASCII character set**, the term "ASCII" being an abbreviation for the **American Standard Code for Information Interchange**. This is one of the

(a)

Bit positions			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0	NUL	DLE	SP	0	@	P	\	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB		7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	–	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	—	o	DEL

(b)



Bit positions			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0					@	P		
0	0	0	1					A	Q		
0	0	1	0					B	R		
0	0	1	1					C	S		
0	1	0	0					D	T		
0	1	0	1					E	U		
0	1	1	0					F	V		
0	1	1	1					G	W		
1	0	0	0					H	X		
1	0	0	1					I	Y		
1	0	1	0					J	Z		
1	0	1	1					K	[		
1	1	0	0					L	\		
1	1	0	1					M	]		
1	1	1	0					N	^		
1	1	1	1					O	—		

Figure 2.6 Two example character sets to produce unformatted text: (a) the basic ASCII character set; (b) supplementary set of mosaic characters.

most widely used character sets and the table includes the binary codewords used to represent each character. As we can see, each character is represented by a unique 7-bit binary codeword. The use of 7 bits means that there are 128 (2^7) alternative characters and the codeword used to identify each character is obtained by combining the corresponding column (bits 7–5) and row (bits 4–1) bits together. Bit 7 is the most significant bit and hence the codeword for uppercase M, for example, is 1001101.

In addition to all the normal alphabetic, numeric and punctuation characters – collectively referred to as **printable characters** – the total ASCII character set also includes a number of **control characters.** These include:

- format control characters: BS (backspace), LF linefeed), CR (carriage return), SP (space), DEL (delete), ESC (escape), and FF (form feed);
- information separators: FS (file separator) and RS (record separator);
- transmission control characters: SOH (start-of-heading), STX (start-of-text), ETX (end-of-text), ACK (acknowledge), NAK (negative acknowledge), SYN (synchronous idle), and DLE (data link escape).

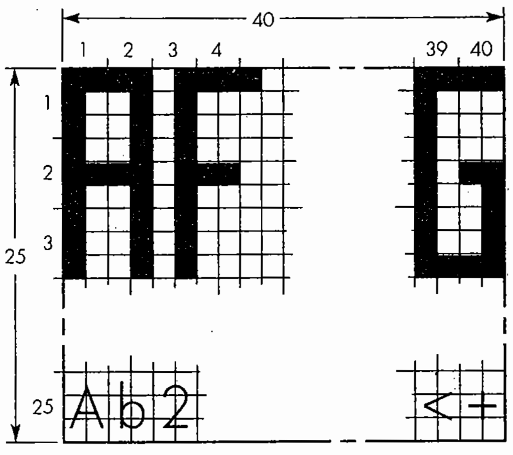
The latter, as we shall describe later in Section 6.4.3, are sometimes used to control the transmission of blocks of characters which are made up of the other characters in the set.

The character set tabulated in Figure 2.6(b) is a supplementary version of that shown in part (a). Here the characters in columns 010/011 and 110/111 are replaced with the set of **mosaic characters** shown. These are then used, together with the various uppercase characters illustrated, to create relatively simple graphical images.

An example application of this particular character set is in **Videotex** and **Teletext** which are general broadcast information services available through a standard television set and used in a number of countries. Some examples of typical Teletext/Videotex symbols are shown in Figure 2.7. As we can see, although in practice the total page is made up of a matrix of symbols and characters which all have the same size (in terms of their height and width), some simple graphical symbols and text of larger sizes can be constructed by the use of groups of the basic symbols.

2.3.2 Formatted text

An example of formatted text is that produced by most word processing packages. It is also used extensively in the publishing sector for the preparation of papers, books, magazines, journals, and so on. It enables documents to be created that consist of characters of different styles and of variable size and shape, each of which can be plain, bold, or italicized. In addition, a variety of document formatting options are supported to enable an author to structure a document into chapters, sections and paragraphs, each with different headings and with tables, graphics, and pictures inserted at appropriate points.



Note: Grid only included as a template.

Figure 2.7 Example Videotex/ Teletext characters.

To achieve each of these features, the author of the document enter
specific command which, typically, results in a defined format-control char
ter sequence – normally a reserved format-control character followed b
pair of other alphabetic or numeric characters – being inserted at the beg
ning of the particular character string, table, graphic or picture. In this w
each page of the document comprises the string of characters that make
the textual part of the page – plus, where appropriate, the associated tabl
graphics, or pictures – with the corresponding format-control charac
sequences interspersed at appropriate points. An example formatted t
string is as shown in part (a) of Figure 2.8 and the printed version of t
string is shown in part(b).

As we can deduce from the above, in order to print a document cons
ing of formatted text, the printer must first be set up – that is,
microprocessor within the printer must be programmed – to detect and in
pret the format-control character sequences in the defined way and
convert the following text, table, graphic, or picture into a line-by-line fo
ready for printing. In addition, to help the author visualize the layout of ea
page prior to printing, commands such as *print preview* are often provid
which cause the page to be displayed on the computer screen in a similar w
to how it will appear when it is printed. This is the origin of the term **WY**
WYG, an acronym for what-you-see-is-what-you-get.

(a)

```
<B><FONT SIZE=4><P>Formatted Text</P>
</B></FONT>
<P>This is an example of formatted text, it includes:</P>
<FONT SIZE=2>
</FONT><I><P>Italics,</I> <B>Bold</B> and <U>Underlining</P>
</U>
<FONT FACE="French Script MT"><P>Different Fonts</FONT> and <FO
SIZE=4>Font Sizes</P>
```

(b)

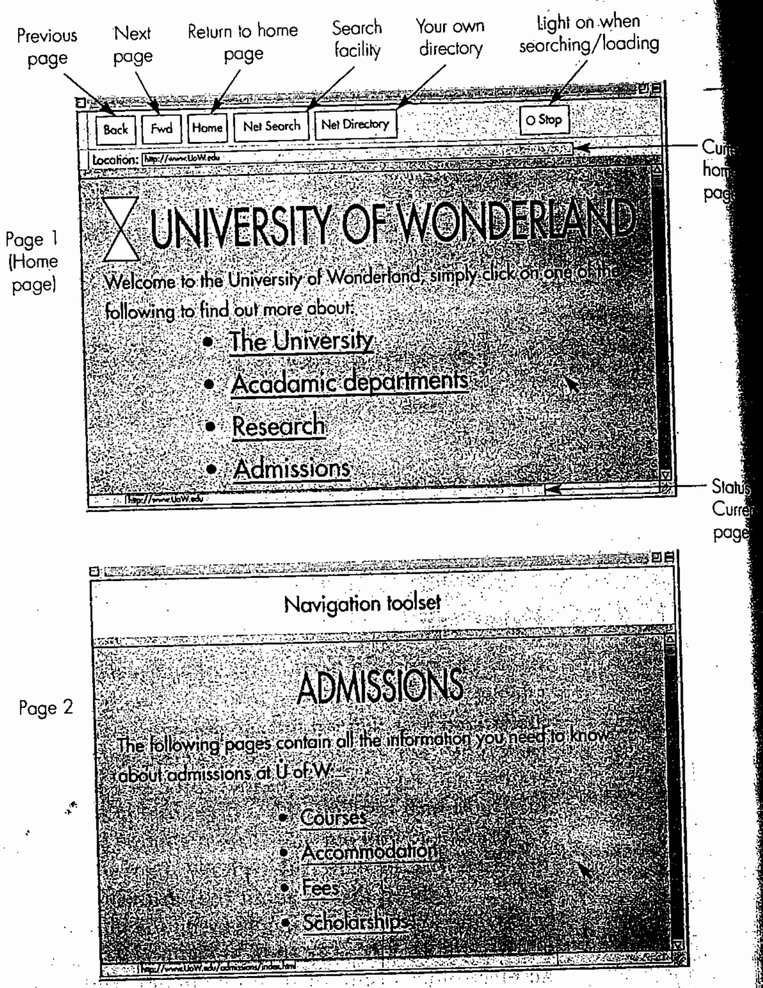
Formatted text
This is an example of formatted text, it includes:
Italics, **Bold** and Underlining
Different Fonts and Font Sizes

Figure 2.8 Formatted text: (a) an example formatted text string; (b) printed version of the string.

2.3.3 Hypertext

As we saw earlier in Section 1.4.2 hypertext is a type of formatted text
enables a related set of documents – normally referred to as **pages** – to be
ated which have defined linkage points – referred to as **hyperlinks** – betw
each other. For example, most universities describe their structure and
courses and support services they offer, in a booklet known as a prospe
Like most such booklets, this is organized in a hierarchical way and, in o
for a reader to find out information about a particular course, and the f
ties offered by the university, typically, the reader would start at the index
use this to access details about the various departments, the courses
offers, and so on, by switching between the different sections of the book

In a similar way, as we show in Figure 2.9, hypertext can be used to cr
an electronic version of such documents with the index, description
departments, courses on offer, library, and other facilities all written in h
text as pages with various defined hyperlinks between them to ena
person to browse through its contents in a user-friendly way. Typically,
linked set of pages that make up the prospectus would all be stored
single server computer. However, should a particular department choo
provide a more in-depth description of the courses and facilities it offers
example, the contents of courses, current research projects, staff profile
publications – these can also be implemented as a linked set of pages on
ferent computer and, providing all the computers at the site are connect
the same network (and use the same set of communication protocols),
tional hyperlinks between the two sets of pages can be introduced.


Previous page
Next page
Return to home page
Search facility
Your own directory
Light on when searching/loading
Back
Fwd
Home
Net Search
Net Directory
O Stop
Location: http://www.UoW.edu
Cu
ho
pag
Page 1 (Home page)
UNIVERSITY OF WONDERLAND
The University
Acadamic departments
Research
Admissions
http://www.UoW.edu
Status
Curre
page
Navigation toolset
Page 2
ADMISSIONS
Courses
Accommodation
Fees
Scholarships
http://www.UoW.edu/admissions/index.html

Welcome to the University of Wonderland, simply click on one of the following to find out more about:

The following pages contain all the information you need to know about admissions at U of W –

Note:
- Page 2 is displayed after clicking the cursor on •Admissions of Page 1
- Selected images can be used as a background.
- Hyperlinks can be either underlined (as shown) or in a different color

Figure 2.9 Example of an electronic document written in hypertext.

The linked set of pages that are stored in a particular server are accesse and viewed using a client program known as a **browser**. This can be runnin in either the same computer on which the server software is running or, mor usually, in a separate remote computer.

Associated with each set of linked pages is a page known as the **home page**. Normally, this comprises a form of index to the set of pages linked to it, each of which has a hyperlink entry-point associated with it. Typically, hyperlinks take the form of an underlined text string and the user initiates the access and display of a particular page by pointing and clicking the mouse on the appropriate string/link.

Associated with each link, in addition to the textual name of the link and the related format-control information for its display, is a unique network-wide name known as a **uniform resource locator** or **URL**. This comprises a number of logical parts – including the unique name of the host computer where the page is stored and also the name of the file containing the page – which collectively enables the browser program to locate and read each requested page. Hence to access the home page of a particular server, the user first enters its URL – in response to a prompt by the browser program – and, in turn, the browser uses this, first to locate the server computer on which the particular page is stored and then to request the page contents from the server. The page contents are stored in a specific formatted text and, on receipt of the contents, the browser displays these on the client computer screen using the included format control commands. In this way, after accessing the home page associated with a site, a user is able to access and browse through the contents of the linked set of pages in the order he or she chooses.

An example of a hypertext language is **HTML** which stands for **HyperText Markup Language**. We shall discuss HTML in detail in section 15.3 and restrict oursleves here to its essential features in order to gain an understanding of the form of representation of a typical hypertext page. HTML is an example of a more general set of what are known as **mark-up languages**. These are used to describe how the contents of a document are to be presented on a printer or a display, the term "mark-up" being that used by a copy editor when the printing of documents was carried out manually. Other mark-up languages include **Postscript** (known as a (printed) page description language), **SGML** (the acronym for **Standard Generalized Mark-up Language** on which HTML is based), **Tex**, and **Latex**. In general, the output of these languages is similar to that produced by many word-processing systems but, unlike word processors, they are concerned only with the formatting of a document in preparation for its printing or display.

As the name implies, HTML is concerned solely with hypertext and has been designed specifically for use with the World Wide Web and, in particular, for the creation of Web pages. It is concerned primarily with the formatting of pages – to enable a browser program running on a remote computer to display a retrieved page on its local screen – and for the specification of hyperlinks – to enable a user to browse interactively through the contents of a set of pages linked together by means of hyperlinks.

The page formatting commands – known as **directives** in HTML and each sandwiched between a pair of **tags** (<>) – include commands to start a new paragraph (<P>), start and end boldface (<B>text</B>), present in the form of a bulleted list (<HL>list</HL>), include an image (<IMG SRC=" an image">), and so on. Other media types such as sound and video clips can also be included, giving rise to the term **hypermedia**. Indeed, the terms "hypermedia" and "hypertext" are often used interchangeably when referring to pages created in HTML.

The specification of a hyperlink is made by specifying both the URL of where the required page is located, together with the textual name of the link. For example, the specification of a hyperlink to a page containing 'Further details' would have the form <A HREF="URL">Further details </A>. Hence, as we can see, a hypertext string is similar to a formatted text string, except that at the linkage points within a page additional text strings are found that define the URL of the linked page.

2.4 Images

Within the context of this book, images include computer-generated images – more generally referred to as **computer graphics** or simply **graphics** – and digitized images of both documents and pictures. Although ultimately all three types of image are displayed (and printed) in the form of a two-dimensional matrix of individual picture elements – known as **pixels** or sometimes **pels** – each type is represented differently within the computer memory or, more generally, in a computer file. Also, each type of image is created differently and hence it is helpful for us to consider each separately.

2.4.1 Graphics

There is a range of software packages and programs available for the creation of computer graphics. These provide easy-to-use tools to create graphics that are composed of all kinds of visual objects including lines, arcs, squares, rectangles, circles, ovals, diamonds, stars, and so on, as well as any form of hand-drawn (normally referred to as **freeform**) objects. Typically, these are produced by drawing the desired shape on the screen by means of a combination of a cursor symbol on the screen – a pencil or paint brush for example – and the mouse. Facilities are also provided to edit these objects – for example to change their shape, size, or color – and to introduce complete predrawn images, either previously created by the author of the graphic or selected from a gallery of images that come with the package. The latter is often referred to as **clip-art** and the better packages provide many hundreds of such images. Textual information can also be included in a graphic, together with precreated tables and graphs and digitized pictures and photographs which have been previously obtained. Objects can overlap each other with a selected object nearer to the front than another. In addition, you can add fill or add shadows to objects in order to give the complete image a three-dimensional (3-D) effect.

A computer's display screen can be considered as being made up of a two-dimensional matrix of individual picture elements – pixels – each of which can have a range of colors associated with it. For example, **VGA (video graphics array)** is a common type of display and, as we show in Figure 2.10(a), consists of a matrix of 640 horizontal pixels by 480 vertical pixels with, for example, 8 bits per pixel which allows each pixel to have one of 256 different colors.

All objects – including freeform objects – are made up of a series of lines that are connected to each other and, what may appear as a curved line, in practice is a series of very short lines each made up of a string of pixels which, in the limit, have the resolution of a pair of adjacent pixels on the screen. Some examples are shown in Figure 2.10(b)

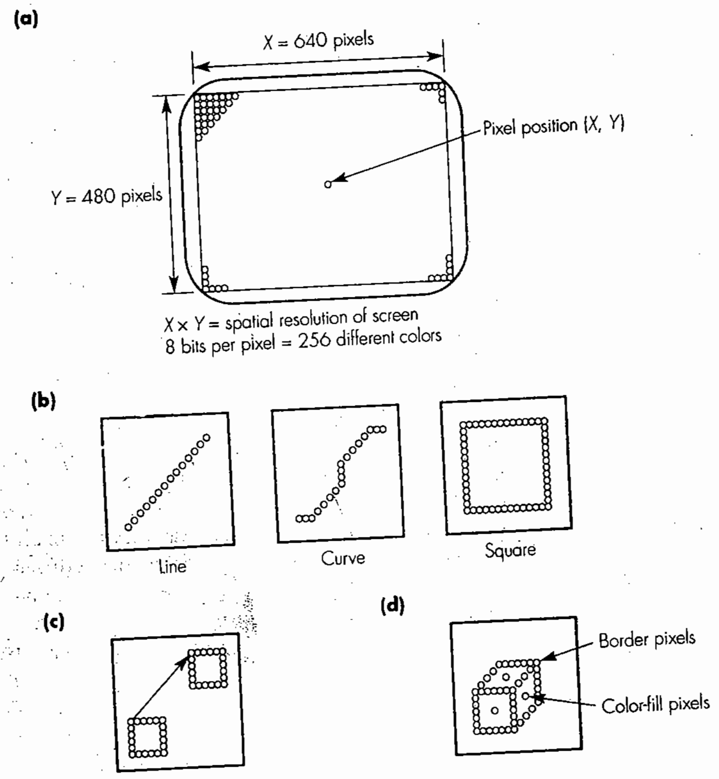


Figure 2.10 Graphics principles: (a) example screen format; (b) som simple object examples; (c) effect of changing position attribute; (d) solid objects.

Each object has a number of **attributes** associated with it. These inclu its shape – a line, a circle, a square, and so on – its size in terms of the pi positions of its border coordinates, the color of the border, its shadow, and on. In this way, editing an object involves simply changing selected attribut associated with the object. For example, as we show in Figure 2.10(c), we move a square to a different location on the screen by simply changing border coordinates and leaving the remaining attributes unchanged.

An object shape is said to be either open or closed. In the case of an ope object, the start of the first line and the end of the last line that make up th object's border are not connected – that is, they do not start and end on th same pixel – whilst with a closed object they are connected. In the case of closed object, the pixels enclosed by its border can all be assigned the sam color – known as the **color-fill** – to create solid objects as shown in Figur 2.10(d). This operation is also known as **rendering**. In this way, all objects ar drawn on the screen by the user simply specifying the name of the object an its attributes – including its color-fill and shadow effect if required – and a s of more basic lower-level commands are then used to determine both th pixel locations that are involved and the color that should be assigned t each pixel.

As we can deduce from the above, the representation of a complet graphic is analogous to the structure of a program written in a high-level programming language. For instance, a program consists of a main bod together with a number of procedures/functions, each of which has a set o parameters associated with it and performs a specific function. In the sam way, a graphic consists of the set of commands (each with attributes) that ar necessary to draw the different objects that make up the graphic. Also, in th same way that the procedures/functions in a program can be a mix of thos created by the writer of the program and those available as library procedures/functions, so the objects associated with a graphic can be either thos created by the author or those selected from the set of standard objects o the clip-art gallery. And in the same way that a procedure/function in a pr gram may, in turn, call a number of lower-level functions, so the comman associated with objects use the lower-level commands to display the objects o the screen. Finally, in the same way that the main body of a program is co cerned with invoking the various procedures/functions in the ord necessary to implement a particular computational task, so the main body a graphic representation is concerned with invoking the different object commands in the correct sequence to create the desired graphic taking int account any overlapping objects.

We can conclude that there are two forms of representation of a computer graphic: a high-level version (similar to the source code of a high-leve program) and the actual pixel-image of the graphic (similar to the byte-strin corresponding to the low-level machine code of the program and known more generally as the **bit-map format**). A graphic can be transferred over network in either form. In general, however, the high-level language form

much more compact and requires less memory to store the image and less bandwidth for its transmission. In order to use the high-level language form, however, the destination must of course be able to interpret the various high-level commands. So instead the bit-map form is often used and, to help with this, there are a number of standardized forms of representation such as **GIF** (**graphical interchange format**) and **TIFF** (**tagged image file format**). There are also software packages such as **SRGP** (**simple raster graphics package**) which convert the high-level language form into a pixel-image form. We shall discuss a selection of these in the next chapter.

2.4.2 Digitized documents

An example of a digitized document is that produced by the scanner associated with a **facsimile (fax) machine**, the principles of which are shown in Figure 2.11.

The scanner associated with a fax machine operates by scanning each complete page from left to right to produce a sequence of scan lines that start

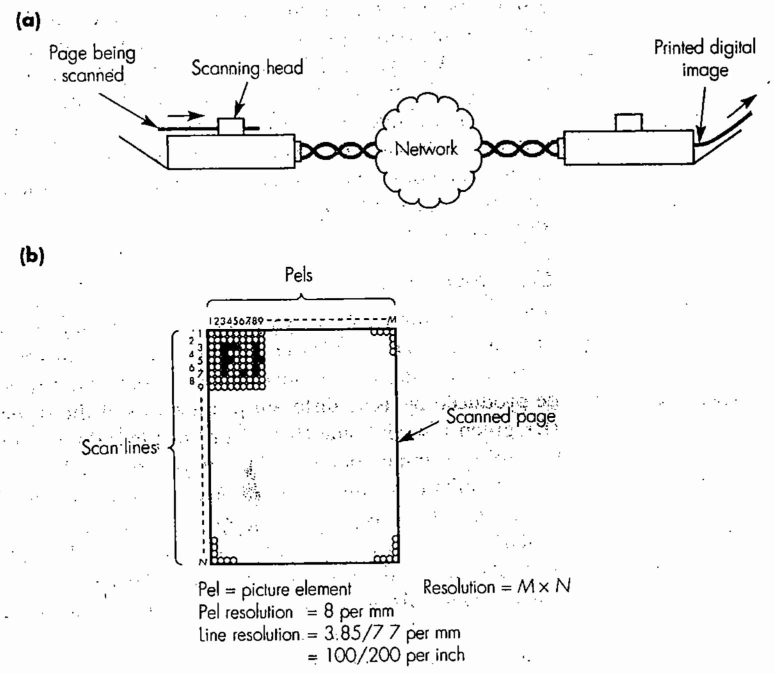


Figure 2.11 Facsimile machine principles: (a) schematic; (b) digitization format.

at the top of the page and end at the bottom. The vertical resolution of the scanning procedure is either 3.85 or 7.7 lines per millimeter which is equivalent to approximately 100 or 200 lines per inch. As each line is scanned, the output of the scanner is digitized to a resolution of approximately 8 picture elements – known as **pels** with fax machines – per millimeter.

Fax machines use just a single binary digit to represent each pel, a 0 for a white pel and a 1 for a black pel. Hence the digital representation of a scanned page is as shown in Figure 2.11(b) which, for a typical page, produces a stream of about two million bits. The printer part of a fax machine then reproduces the original image by printing out the received stream of bits to a similar resolution. In general, the use of a single binary digit per pel means that fax machines are best suited to scanning bitonal (black-and-white) images such as printed documents comprising mainly textual information.

2.4.3 Digitized pictures

In the case of scanners which are used for digitizing continuous-tone monochromatic images – such as a printed picture or scene – normally, more than a single bit is used to digitize each picture element. For example, good quality black-and-white pictures can be obtained by using 8 bits per picture element. This yields 256 different levels of gray per element – varying between white and black – which gives a substantially improved picture quality over a facsimile image when reproduced. In the case of color images, in order to understand the digitization format used, it is necessary first to obtain an understanding of the principles of how color is produced and how the picture tubes used in computer monitors (on which the images are eventually displayed) operate.

Color principles

It has been known for many years that the human eye sees just a single color when a particular set of three primary colors are mixed and displayed simultaneously. In fact, a whole spectrum of colors – known as a **color gamut** – can be produced by using different proportions of the three primary colors red (R), green (G), and blue (B). This principle is shown in Figure 2.12 together with some examples of colors that can be produced.

The mixing technique used in part (a) is known as **additive color mixing** which, since black is produced when all three primary colors are zero, is particularly useful for producing a color image on a black surface as is the case in display applications. It is also possible to perform the complementary **subtractive color mixing** operation to produce a similar range of colors. This is shown in part(b) of the figure and, as we can see, with subtractive mixing white is produced when the three chosen primary colors cyan (C), magenta (M), and yellow (Y) are all zero. Hence this choice of colors is particularly useful for producing a color image on a white surface as is the case in printing applications.

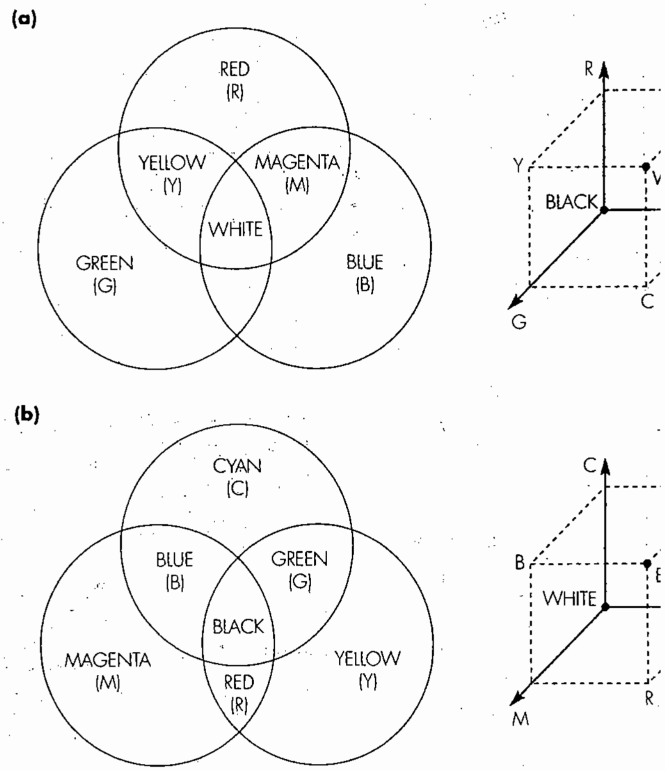


Figure 2.12 Color derivation principles: (a) additive color mix
(b) subtractive color mixing.

The same principle is used in the picture tubes associated with
vision sets with the three primary colors R, G, and B. Also, in most
monitors since, in general, those used with personal computers us
picture tubes as are used in television sets. Hence, in order to be
with the computer monitors on which digital pictures are norma
the digitization process used yields a color image that can be d
played on the screen of either a television set or a computer mo
general principles associated with the process are shown in Figure

Raster-scan principles

The picture tubes used in most television sets operate using what i
a **raster**-scan. This involves a finely-focussed electron beam – t
being scanned over the complete screen. Each complete scan c
number of discrete horizontal lines the first of which starts at t

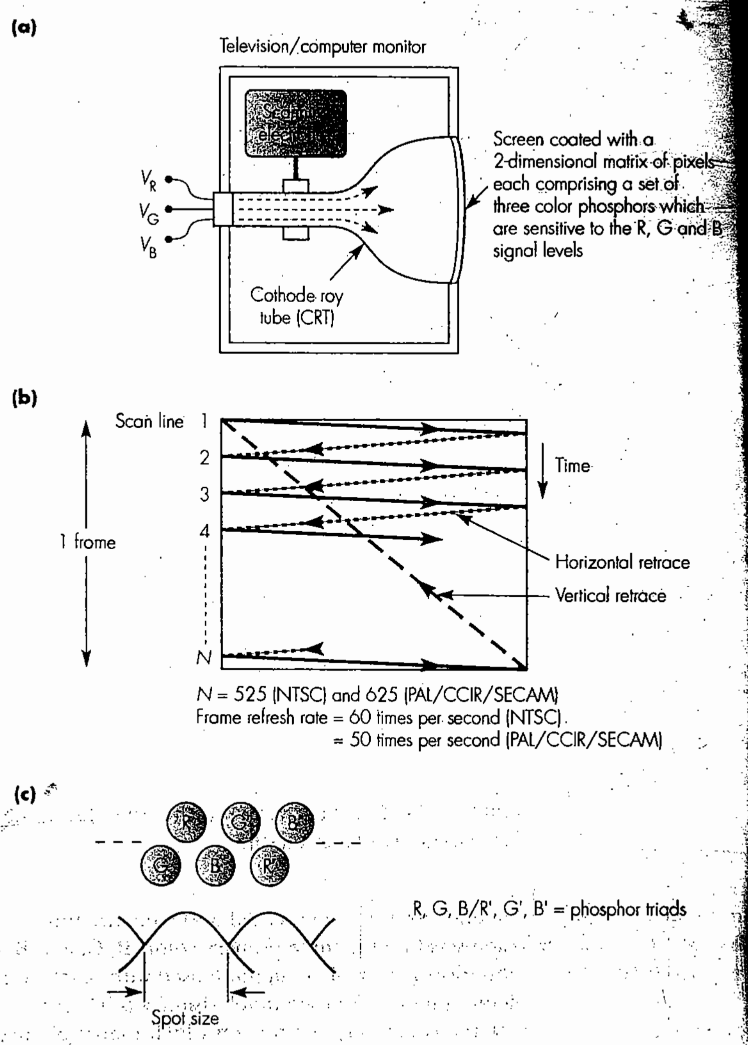

(a)
Television/computer monitor
V_R
V_G
V_B
Screen coated with a 2-dimensional matrix of pixels each comprising a set of three color phosphors which are sensitive to the R, G and B signal levels
Cathode ray tube (CRT)
(b)
Scan line
1
2
3
4
N
1 frame
Time
Horizontal retrace
Vertical retrace
N = 525 (NTSC) and 625 (PAL/CCIR/SECAM)
Frame refresh rate = 60 times per second (NTSC)
= 50 times per second (PAL/CCIR/SECAM)
(c)
R
G
B
G'
B'
R'
R, G, B/R', G', B' = phosphor triads
Spot size

Figure 2.13 Television/computer monitor principles: (a) schematic; (b) raster-scan principles; (c) pixel format on each scan line.

corner of the screen and the last of which ends at the bottom right corner. At this point the beam is deflected back again to the top left corner and the scanning operation repeats in the same way. This type of scanning is called **progressive scanning** and is shown in diagrammatic form in Figure 2.13(b).

Each complete set of horizontal scan lines is called a **frame** and, as we can see, each frame is made up of *N* individual scan lines where *N* is either 525 (North and South America and most of Asia) or 625 (Europe and a number of other countries). The inside of the display screen of the picture tube is coated with a light-sensitive phosphor that emits light when energized by the electron beam. The amount of light emitted – its brightness – is determined by the power in the electron beam at that instant. During each horizontal (line) and vertical (frame) retrace period the electron beam is turned off and, to create an image on the screen, the level of power in the beam is changed as each line is scanned.

In the case of black-and-white picture tubes just a single electron beam is used with a white-sensitive phosphor. Color tubes use three separate, closely-located beams and a two-dimensional matrix of pixels. Each pixel comprises a set of three related color-sensitive phosphors, one each for the red, green, and blue signals. The set of three phosphors associated with each pixel is called a **phosphor triad** and a typical arrangement of the triads on each scan line is as shown in Figure 2.13(c). As we can deduce from this, although in theory each pixel represents an idealized rectangular area which is independent of its neighboring pixels, in practice each pixel has the shape of a **spot** which merges with its neighbors. A typical spot size is 0.025 inches (0.635 mm) and, when viewed from a sufficient distance, a continuous color image is seen.

Television picture tubes were designed, of course, to display moving images. The persistence of the light/color produced by the phosphor, therefore, is designed to decay very quickly and hence it is necessary to continuously **refresh** the screen. In the case of a moving image, the light signals associated with each frame change to reflect the motion that has taken place during the time required to scan the preceding frame, while for a static/still image, the same set of light signals are used for each frame. The **frame refresh rate** must be high enough to ensure the eye is not aware the display is continuously being refreshed. A low refresh rate leads to what is called **flicker** which is caused by the previous image fading from the eye retina before the following image is displayed. To avoid this, a refresh rate of at least 50 times per second is required. In practice, the frame refresh rate used is determined by the frequency of the mains electricity supply which is either 60 Hz in North and South America and most of Asia and 50 Hz in Europe and a number of other countries.

Most current picture tubes operate in an analog mode, that is, the amplitude of each of the three color signals is continuously varying as each line is scanned. In the case of digital television – and digitized pictures stored within the memory of a computer – the color signals are in a digital form and comprise a string of pixels with a fixed number of pixels per scan line. Hence in order to display a stored image, the pixels that make up each line are read from memory in time-synchronism with the scanning process and converted into a continuously varying analog form by means of a digital-to-analog converter.

Since in practice the area of the computer memory that holds the string
pixels that make up the image – the pixel image – must be accessed contin
ously as each line is scanned, normally a separate block of memory known
the **video RAM** – RAM being the acronym for **random access memory** –
used to store the pixel image. In this way, the graphics program needs only
write into the video RAM whenever either selected pixels or the total imag
changes. An example architecture showing the various steps involved is give
in Figure 2.14

Typically, the graphics program is used to create the high-level version
the image interactively (using either the keyboard or a mouse) and the **di**
play controller part of the program interprets sequences of displa
commands and converts them into displayed objects by writing the appropr
ate pixel values into the video RAM. The latter is also known, therefore,
the **frame/display/refresh buffer**. Normally the **video controller** is a hard
ware subsystem that reads the pixel values stored in the video RAM i
time-synchronism with the scanning process and, for each set of pixel values
converts these into the equivalent set of red, green, and blue analog signal
for output to the display.

Pixel depth

The number of bits per pixel is known as the **pixel depth** and determines th
range of different colors that can be produced. Examples are 12 bits – 4 bits
per primary color yielding 4096 different colors – and 24 bits – 8 bits pe

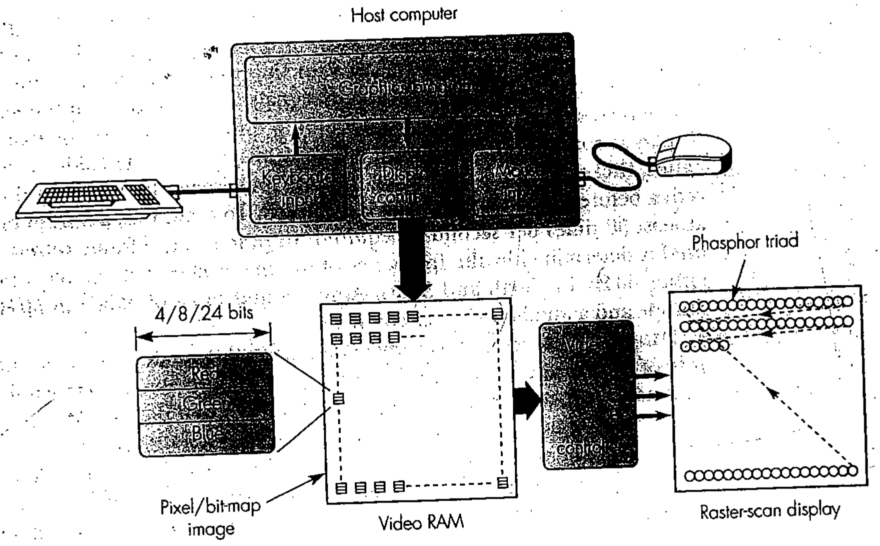


Figure 2.14 Raster-scan display architecture.

primary color yielding in excess of 16 million (2^{24}) colors. In practice, how-
ever, the eye cannot discriminate between such a range of colors and so in
some instances a selected subset of this range is used. The selected colors in
the subset are then stored in a table and each pixel value is used as an address
to a location within the table which contains the corresponding three color
values. The table is known as the **color look-up table** or **CLUT**. For example
if each pixel is 8 bits and the CLUT contains 24 bit entries, this will provide
subset of 256 (2^8) different colors selected from a palette of 16 million (2^{24}
colors. In this way, the amount of memory required to store an image can b
reduced significantly.

Aspect ratio

Both the number of pixels per scanned line and the number of lines pe
frame vary, the actual numbers used being determined by what is known
the **aspect ratio** of the display screen. This is the ratio of the screen width
the screen height. The aspect ratio of current television tubes is 4/3 wi
older tubes – on which PC monitors are based – and 16/9 with the wid
screen television tubes.

In the United States, the standard for color television has been defin
by the **National Television Standards Committee (NTSC)** while in Euro
three color standards exist **PAL** (UK), **CCIR** (Germany), and **SECA**
(France). As we indicated earlier, the NTSC standard uses 525 scan lines
frame while the three European standards all use 625 scan lines. In neit
case, however, are all lines displayed on the screen since some are used
carry control and other information. In practice, therefore, the numbe
visible lines per frame – which is equal to the vertical resolution in term
pixels – is 480 with an NTSC monitor and 576 with the other three standa
Thus in order to avoid distortion on a screen which has a 4/3 aspect rat
for example when displaying a square of, say, ($N \times N$) pixels – it is neces
to have 640 pixels (480 × 4/3) per line with an NTSC monitor and
(576 × 4/3) pixels per line with a European monitor. This produces a la
structure that is said to produce **square pixels** and is shown in diagramn
form in Figure 2.15. Some example screen resolutions associated with
more common computer monitors based on television picture tub
together with the amount of memory required to store the correspon
image – are shown in Table 2.1

As we can deduce from the table, the memory requirements to st
single digital image can be high and vary between 307.2 kbytes for an i
displayed on a **VGA (video graphics array)** screen with 8 bits per
through to approximately 2.36 Mbytes for a **SVGA (Super VGA)** scree
24 bits per pixel. It should be noted that the more expensive computer
tors are not based on television picture tubes and hence are not const
by the 4/3 aspect ratio. An example is 1280 × 1024 × 24 which may
refresh rate as high as 75 frames per second to produce a sharper imag

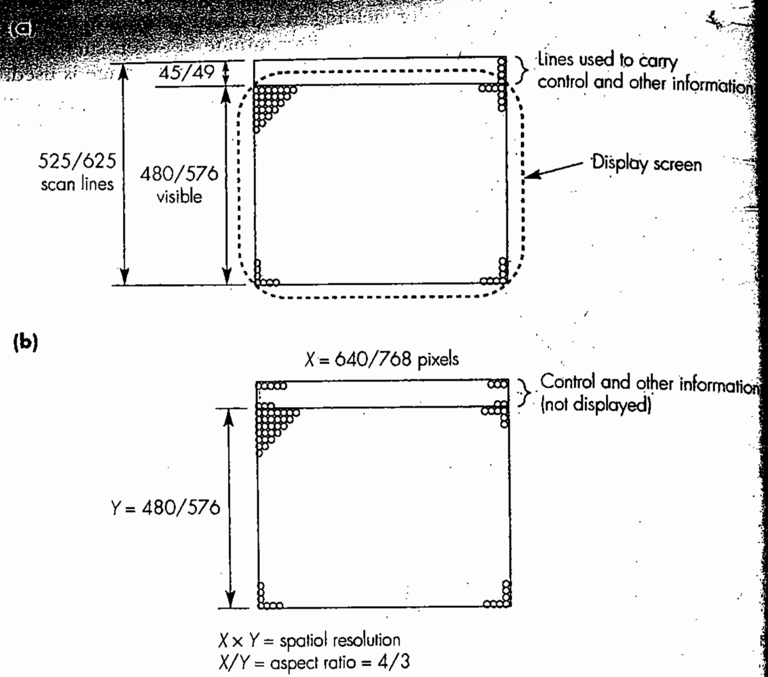

(a)
45/49
Lines used to carry control and other information
525/625 scan lines
480/576 visible
Display screen
(b)
X = 640/768 pixels
Control and other information (not displayed)
Y = 480/576
X × Y = spatial resolution
X/Y = aspect ratio = 4/3

Figure 2.15 Screen resolutions: (a) visible lines per frame; (b) digitization spatial resolution.

Table 2.1 Example display resolutions and memory requirements.

Standard	*Resolution*	*Number of colors*	*Memory required per frame (bytes)*
VGA	640 × 480 × 8	256	[illegible]
XGA	640 × 480 × 16	64K	[illegible]
	1024 × 768 × 8	256	[illegible]
SVGA	800 × 600 × 16	64K	[illegible]
	1024 × 768 × 8	256	[illegible]
	1024 × 768 × 24	16M	[illegible]

Example 2.3

Derive the time to transmit the following digitized images at both 64 kbps and 1.5 Mbps:

- a 640 × 480 × 8 VGA-compatible image,
- a 1024 × 768 × 24 SVGA-compatible image.

Answer:

The size of each image in bits is:

VGA = 640 × 480 × 8 = 2.4576 Mbits

SVGA = 1024 × 768 × 24 = 18.874368 Mbits

Hence the time to transmit each image is:

$$\text{At 64 kbps: VGA} = \frac{2.4576 \times 10^6}{64 \times 10^3} = 38.4\text{s}$$

$$\text{SVGA} = \frac{18.874368 \times 10^6}{64 \times 10^3} = 294.912\text{s}$$

$$\text{At 1.5 Mbps: VGA} = \frac{2.4576 \times 10^6}{1.5 \times 10^6} = 1.6384\text{s}$$

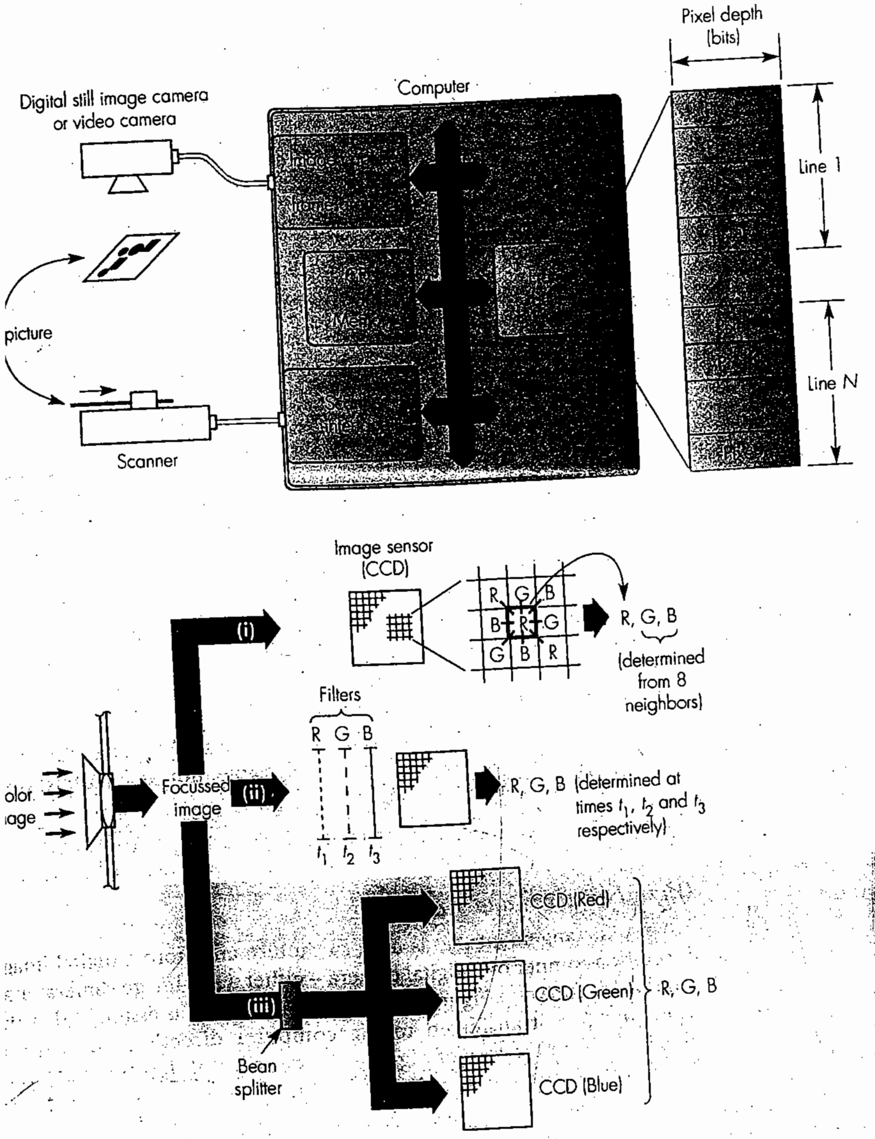
$$\text{SVGA} = \frac{18.874368 \times 10^6}{1.5 \times 10^6} = 12.5829\text{s}$$

As we can see, the times to transmit a signal image at 64 kbps are such that interactive access would not be feasible, nor at 1.5 Mbps with the higher-resolution SVGA image.

Digital cameras and scanners

A typical arrangement that is used to capture and store a digital image produced by a scanner or a digital camera – either a still-image camera or a video camera – is shown in Figure 2.16(a). In the figure it is assumed that the captured image is transferred to the computer directly as it is produced. Alternatively, in the case of digital cameras, a set of digitized images can be stored within the camera itself and then downloaded into the computer at a later time.

An image is captured within the camera/scanner using a solid-state device called an **image sensor**. This is a silicon chip which, in digital cameras, consists of a two-dimensional grid of light-sensitive cells called **photosites**. When the camera shutter is activated, each photosite stores the level of intensity of the light that falls on it. A widely-used image sensor is a **charge-coupled**



2.16 **Color image capture: (a) schematic; (b) RGB signal generation alternatives.**

device (CCD). This comprises an array of photosites on its surface and operates by converting the level of light intensity that falls on each photosite into an equivalent electrical charge. The level of charge – and hence light intensity – stored at each photosite position is then read out and converted into a digital value using an ADC. A similar technique is used in scanners except the image sensor comprises just a single row of photosites. These are exposed in time-sequence with the scanning operation and each row of stored charges are read out and digitized before the next scan occurs.

For color images, the color associated with each photosite – and hence pixel position – is obtained in a number of ways. These include the three methods shown in Figure 2.16(b).

(i) In this method, the surface of each photosite is coated with either a red, green, or blue filter so that its charge is determined only by the level of red, green, or blue light that falls on it. The coatings are arranged in a 3 × 3 grid structure as shown in the figure. The color associated with each photosite/pixel is then determined by the output of the photosite – R, G, or B – together with each of its 8 immediate neighbors. The levels of the two other colors in each pixel are then estimated by an interpolation procedure involving all nine values. This method is used with most consumer-quality cameras.

(ii) This method involves the use of three separate exposures of a single image sensor, the first through a red filter, the second a green filter, and the third a blue filter. The color associated with each pixel position is then determined by the charge obtained with each of the three filters – R, G, and B. Since three separate exposures are required for each image, this approach cannot be used with video cameras. It is used primarily with high-resolution still-image cameras in locations such as photographic studios where the camera can be attached to a tripod.

(iii) This method uses three separate image sensors, one with all the photosites coated with a red filter, the second coated with a green filter, and the third coated with a blue filter. A single exposure is used with the incoming light split into three beams each of which exposes a separate image sensor. This method is used in professional-quality high-resolution still and moving image cameras since, in general, they are more costly owing to the use of three separate sensors and associated signal processing circuits.

Once each image/frame has been captured and stored on the image sensor(s), the charge stored at each photosite location is read and digitized. Using a CCD, the set of charges on the matrix of photosites are read a single row at a time. First the set of charges on the first row of photosites are transferred to what is called the **readout register**. Each of the photosites in a row is coupled to the corresponding photosites in the two adjoining rows and, as each row is transferred to the readout register, the set of charges on each of the other rows in the matrix move down to the next row of photosite positions. Once in the readout register, the charge on each photosite position is shifted out, amplified and digitized using an ADC. This procedure then repeats until the set of charges on all rows have been read out and digitized.

For a low-resolution image of 640 × 480 pixels and a pixel depth of 24 bits – 8 bits each for the R, G, and B signals – the amount of memory required to store each image is 921 600 bytes. If this is output directly to the computer, then the bit-map can be loaded straight into the frame buffer ready to be displayed. If it is to be stored within the camera, however, then multiple images of this size need to be stored prior to them being output to a computer. The set of images are stored in an integrated circuit memory that is either on a removable card or fixed within the camera. In the first case, the card is simply removed and inserted into the PCMCIA slot of a computer and in the second case the contents of the memory are downloaded to the computer by means of a cable link. Once within the computer, software can be used to insert the digital image(s) into a document, send it by email, and so on. Alternatively, photo-editing software can be used to manipulate a stored image; for example, to change its size or color

There are a number of file formats used to store a set of images. One of the most popular is a version of the tagged image file format (TIFF) called **TIFF for electronic photography (TIFF/EP)**. This allows many different types of image data to be stored in the image file including data (such as the date and time and various camera settings) associated with each image.

2.5 Audio

Essentially, we are concerned with two types of audio signal: speech signals as used in a variety of interpersonal applications including telephony and video telephony, and music-quality audio as used in applications such as CD-on-demand and broadcast television. In general, audio can be produced either naturally by means of a microphone or electronically using some form of synthesizer. In the case of a synthesizer, the audio is created in a digital form and hence can be readily stored within the computer memory. A microphone, however, generates a time-varying analog signal and in order to store such signals in the memory of a computer, and to transmit them over a digital network, they must first be converted into a digital form using an audio signal encoder. Also, since loudspeakers operate using an analog signal, on output of all digitized audio signals the stream of digitized values must be converted back again into its analog form using an audio signal decoder.

We discussed the general principles behind the design of a signal encoder and decoder earlier in the chapter in Section 2.2 and here we will simply apply these principles to explain the digitization of both speech and music produced by a microphone. We shall then discuss the format of synthesized audio in a separate section.

The bandwidth of a typical speech signal is from 50 Hz through to 10 kHz and that of a music signal from 15 Hz through to 20 kHz. Hence the sampling rate used for the two signals must be in excess of their Nyquist rate which is 20 ksps (2 × 10 kHz) for speech and 40 ksps (2 × 20 kHz) for music. The number of bits per sample must be chosen so that the quantization noise generated by the sampling process is at an acceptable level relative to the minimum signal level. In the case of speech, assuming linear (equal) quantization intervals, tests have shown that this dictates the use of a minimum of 12 bits per sample and for music 16 bits. In addition, since in most applications involving music stereophonic (stereo) sound is utilized (and hence two such signals must be digitized) this results in a bit rate double that of a monaural (mono) signal.

Example 2.4

Assuming the bandwidth of a speech signal is from 50 Hz through to 10 kHz and that of a music signal is from 15 Hz through to 20 kHz, derive the bit rate that is generated by the digitization procedure in each case assuming the Nyquist sampling rate is used with 12 bits per sample for the speech signal and 16 bits per sample for the music signal. Derive the memory required to store a 10 minute passage of stereophonic music.

Answer:

(i) Bit rates: Nyquist sampling rate = 2 f_{max}

Speech: Nyquist rate = 2 × 10 kHz = 20 kHz or 20 ksps

Hence with 12 bits per sample, bit rate generated = 20k × 12 = 240 kbps

Music: Nyquist rate = 2 × 20 kHz = 40 kHz or 40 ksps

Hence bit rate generated = 40k × 16 = 640 kbps (mono)

or 2 × 640k = 1280 kbps (stereo)

(ii) Memory required: Memory required = bit rate (bps) × time (s)/8 bytes

Hence at 1280 kbps and 600 s,

$$\text{Memory required} = \frac{1280 \times 10^3 \times 600}{8} = 96\ \text{Mbytes}$$

In practice, both the sampling rate used and the number of bits per sample are often less than these values. In the case of speech, for example, the bandwidth of the network used in many interpersonal applications is often much less than the bandwidth of the source signal thus dictating a lower sampling rate with fewer bits per sample. Similarly, with music, the sampling rate is often lowered in order to reduce the amount of memory that is required to store a particular passage of music. A practical example of the digitization parameters used in each case is now presented.

2.5.1 PCM speech

As we described in Chapter 1, most interpersonal applications involving speech use for communication purposes a public switched telephone network (PSTN). Because this has been in existence for many years the operating parameters associated with it were defined some time ago. Initially, a PSTN operated with analog signals throughout, the source speech signal being transmitted and switched (routed) unchanged in its original analog form. Progressively, however, the older analog transmission circuits were replaced by digital circuits. This was carried out over a number of years and, because of the need to interwork between the earlier analog and newer digital equipments during the transition period, the design of the digital equipment was based on the operating parameters of the earlier analog network. The bandwidth of a speech circuit in this network was limited to 200 Hz through to 3.4 kHz. Also, although the Nyquist rate is 6.8 kHz, the poor quality of the bandlimiting filters used meant that a sampling rate of 8 kHz was required to avoid aliasing. In order to minimize the resulting bit rate, 7 bits per sample were selected for use in North America and Japan and 8 bits per sample in Europe (both including a sign bit) which, in turn, yields bit rates of 56 kbps and 64 kbps respectively. More modern systems have moved to using 8 bits per sample in each case, giving a much improved performance over early 7 bit systems. The digitization procedure is known as **pulse code modulation** or **PCM** and the international standard relating to this is defined in **ITU-T Recommendation G.711**. Figure 2.17 (a) shows the circuits that make up a PCM encoder and decoder.

As we can see, both the encoder and decoder, in addition to the circuits shown earlier in Figs. 2.2(a) and 2.5(a), consist of two additional circuits: a **compressor** (encoder) and an **expander** (decoder). The role of these circuits can best be described by reconsidering the quantization operation described earlier. This used equal – also known as linear – quantization intervals which means that, irrespective of the magnitude of the input signal, the same level of quantization noise is produced. The effect of this is that the noise level is the same for both low amplitude (quiet) signals and high amplitude (loud) signals. The ear, however, is more sensitive to noise on quiet signals than it is on loud signals. Hence to reduce the effect of quantization noise with just 8 bits per sample, in a PCM system the quantization intervals are made non-linear (unequal) with narrower intervals used for smaller amplitude signals than for larger signals. This is achieved by means of the compressor circuit and, at the destination, the reverse operation is performed by the expander circuit. The overall operation is known as **companding**. The input/output relationship of both circuits is shown in Figure 2.17(b) and (c) respectively; that shown in part (b) is known as the **compression characteristic** and that in part (c) the **expansion characteristic**. For clarity, just 5 bits per sample are used.

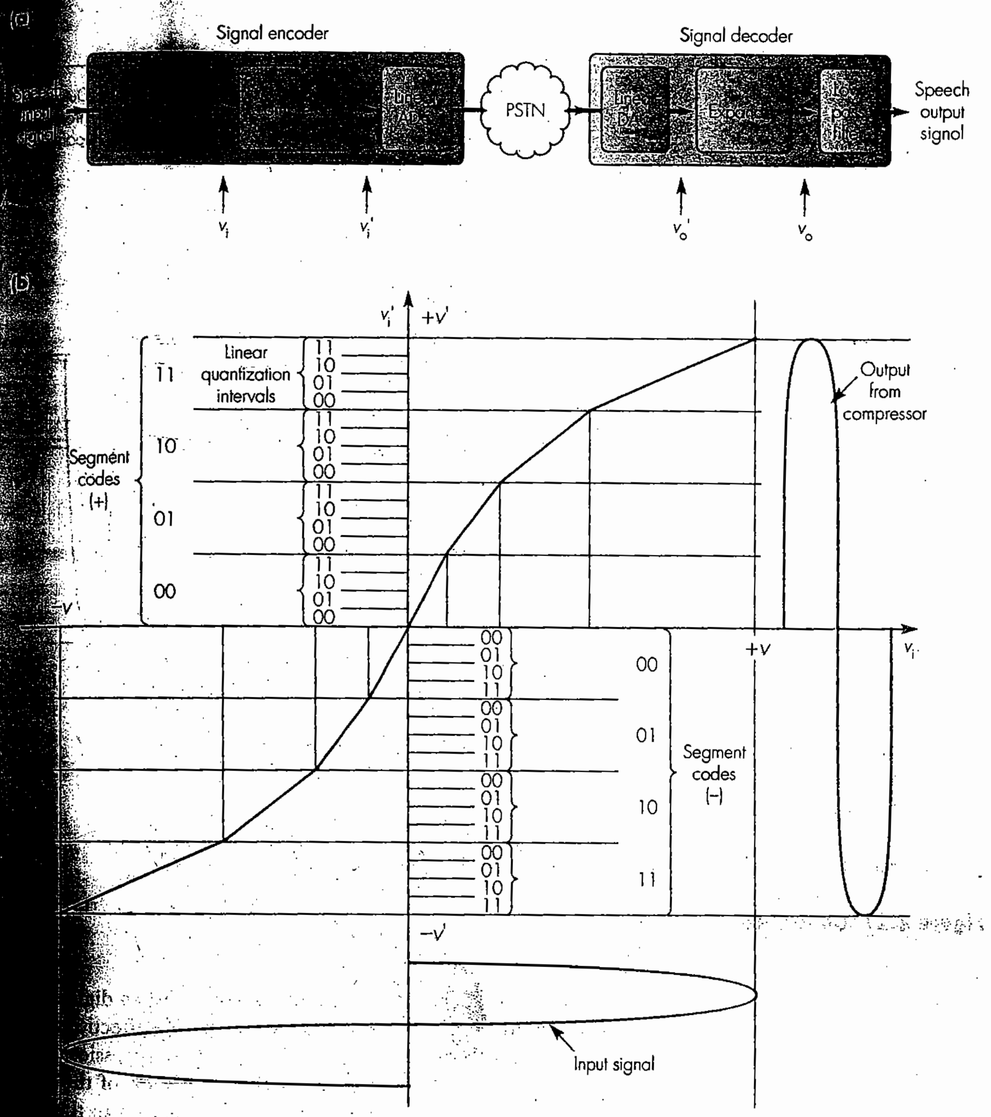
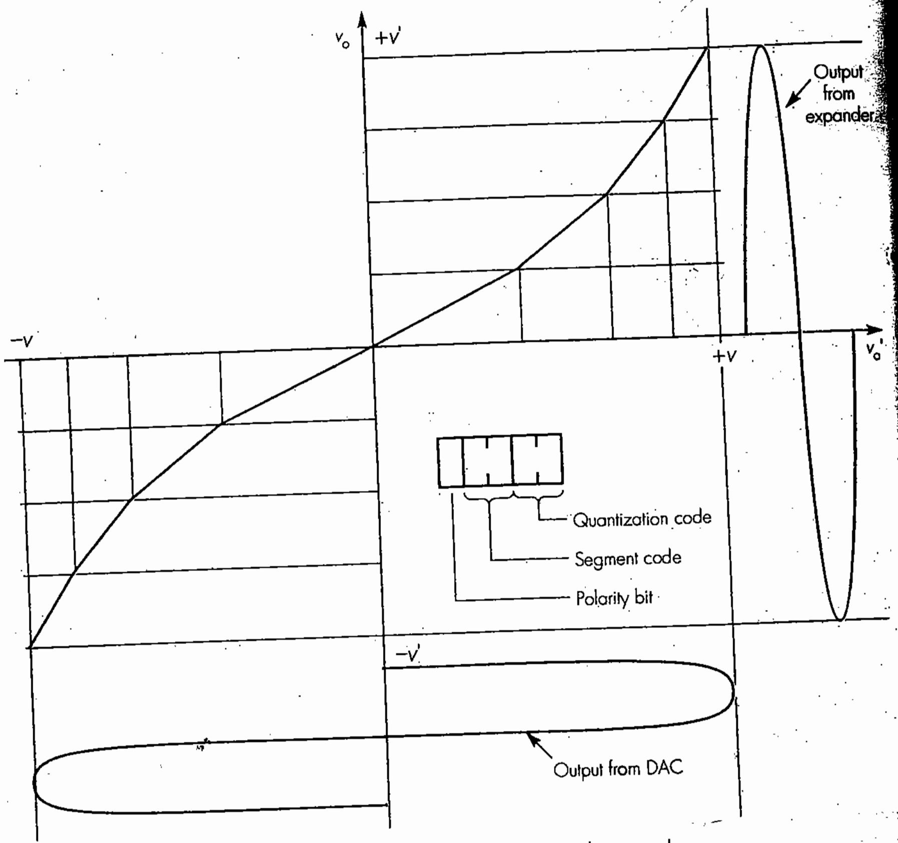


Figure 2.17 PCM principles: (a) signal encoding and decoding schematic; (b) compressor characteristic; (c) expander characteristic.



lote that in the G.711 standard a 3-bit segment code and 4-bit quantization code are used.

e 2.17 Continued

Prior to the input signal being sampled and converted into a digital by the ADC, it is passed through the compressor circuit which effectively presses the amplitude of the input signal. The level of compression – hence the quantization intervals – increases as the amplitude of the i signal increases. The resulting compressed signal is then passed to the which, in turn, performs a linear quantization on the compressed sig Similarly, at the destination, each received codeword is first fed into a li DAC. The analog output from the DAC is then passed to the expander ci which performs the reverse operation of the compressor circuit. M modern systems perform both the compressor and expander operations tally but the same principles apply.

In practice, for historical reasons, there are two different compression-expansion characteristics in use: **μ-law**, which is used in North America and Japan, and **A-law** which is used in Europe and some other countries. Hence, as we can deduce from Figure 2.17(a), it is necessary to carry out a conversion operation when communicating between the two systems. In both cases, however, the use of companding gives a perceived level of performance with 8 bits that is comparable with the performance obtained with 12 bits and uniform quantization intervals.

2.5.2 CD-quality audio

The discs used in CD players and CD-ROMs are digital storage devices for stereophonic music and more general multimedia information streams. There is a standard associated with these devices which is known as the **CD-digital audio (CD-DA)** standard. As indicated earlier, music has an audible bandwidth of from 15 Hz through to 20 kHz and hence the minimum sampling rate is 40 ksps. In the standard, however, the actual rate used is higher than this rate firstly, to allow for imperfections in the bandlimiting filter used and secondly, so that the resulting bit rate is then compatible with one of the higher transmission channel bit rates available with public networks.

One of the sampling rates used is 44.1 ksps which means the signal is sampled at 23 microsecond intervals. Since the bandwidth of a recording channel on a CD is large, a high number of bits per sample can be used. The standard defines 16 bits per sample which, as indicated earlier, tests have shown to be the minimum required with music to avoid the effect of quantization noise. With this number of bits, linear quantization can be used which yields 65 536 equal quantization intervals. The recording of stereophonic music requires two separate channels and hence the total bit rate required is double that for mono. Hence:

$$\text{Bit rate per channel} = \text{sampling rate} \times \text{bits per sample}$$
$$= 44.1 \times 10^3 \times 16 = 705.6\,\text{kbps}$$

and, $$\text{Total bit rate} = 2 \times 705.6 = 1.411\,\text{Mbps}$$

This is also the bit rate used with CD-ROMs which are widely used for the distribution of multimedia titles. Within a computer, however, in order to reduce the access delay, multiples of this rate are used.

As we can deduce from Example 2.5, it is not feasible to interactively access a 30s portion of a multimedia title over a 64 kbps channel. And with a 1.5 Mbps channel the time is still too high for interactive purposes.

2.5.3 Synthesized audio

Once digitized, any form of audio can be stored within a computer. However, as we can see from the results obtained in the next example, the amount of

mple 2.5

Assuming the CD-DA standard is being used, derive:

(i) the storage capacity of a CD-ROM to store a 60 minute multim
title,

(ii) the time to transmit a 30 second portion of the title using a tra
mission channel of bit rate:

- ■ 64 kbps
- ■ 1.5 Mbps.

Answer:

(i) The CD-DA digitization procedure yields a bit rate of 1.411 Mbps
Hence storage capacity for 60 minutes

$= 1.411 \times 60 \times 60$ Mbits

$= 5079.6$ Mbits or 634.95 Mby

(ii) One 30 second portion of the title $= 1.411 \times 30 = 42.33$ Mbits
Hence time to transmit this data:

$$\text{At 64 kbps} = \frac{42.33 \times 10^6}{64 \times 10^3} = 661.4 \text{ s (about 11 minutes)}$$

$$\text{At 1.5 Mbps} = \frac{42.33 \times 10^6}{1.5 \times 10^6} = 28.22$$

memory required to store a digitized audio waveform can be very large, eve for relatively short passages. It is for this reason that synthesized audio is ofte used in multimedia applications since the amount of memory required ca be between two and three orders of magnitude less than that required t store the equivalent digitized waveform version. In addition, it is much easie to edit synthesized audio and to mix several passages together. The mai components that make up an audio synthesizer are shown in Figure 2.18.

The three main components are the computer (with various applicatio programs), the keyboard (based on that of a piano), and the set of soun generators. Essentially, the computer takes input commands from the key board and outputs these to the sound generators which, in turn, produce th corresponding sound waveform – via DACs – to drive the speakers.

Pressing a key on the keyboard of a synthesizer has a similar effect to pressing a key on the keyboard of a computer inasmuch as, for each key that is pressed, a different codeword – known as a message with a synthesizer key- board – is generated and read by the computer program. Essentially, the messages indicate such things as which key on the keyboard has been pressed and the pressure applied. The control panel contains a range of different

---

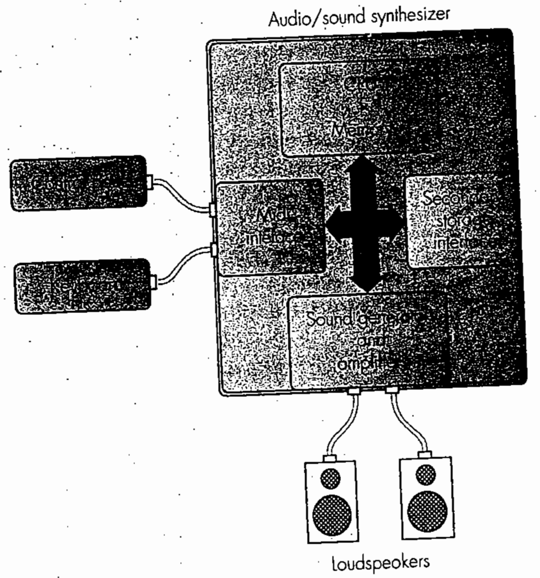


Figure 2.18 Audio/sound synthesizer schematic.

switches and sliders that collectively allow the user to indicate to the computer program additional information such as the volume of the generated output and selected sound effects to be associated with each key.

The secondary storage interface allows the sequence of messages – including those associated with the control panel – relating to a particular piece of audio to be saved on secondary storage such as a floppy disk. In addition, there are programs to allow the user to edit a previously entered passage and, if required, to mix several stored passages together. The sequencer program associated with the synthesizer then ensures that the resulting integrated sequence of messages are synchronized and output to the sound generators to create the merged passage.

As well as a (piano) keyboard, there is a range of other possible inputs from instruments such as an electric guitar, all of which generate messages similar to those produced by the keyboard. Hence in order to discriminate between the inputs from the different possible sources, a standardized set of messages have been defined for both input and for output to the corresponding set of sound generators. These are defined in a standard known as the **Music Instrument Digital Interface (MIDI)**. As the name implies, this does not just define the format of the standardized set of messages used by a synthesizer, but also the type of connectors, cables, and electrical signals that are used to connect any type of device to the synthesizer.

The format of a MIDI message consists of a *status byte*, which define particular event that has caused the message to be generated, followed number of *data bytes* which collectively define a set of parameters associ with the event. An example of an event is a key being pressed on the board and typical parameters would then be the identity of the key, pressure applied, and so on. As we indicated earlier, there can be a variet instrument types used for input and output. Thus it is also necessary to i tify the type of instrument that generated the event so that when corresponding message is output to the sound generators, the appropr type of sound is produced. Thus the different types of device have a M code associated with them; a piano, for example, has a code of 0 and a vio a code of 40. In addition, some codes have been assigned for specific spe effects such as the sound from a cannon and the applause from an audien

As we can deduce from the above, a passage of audio produced by a s thesizer consists of a very compact sequence of messages – each comprisin string of bytes – which can either be played out by the sequencer progr directly – and hence heard by the composer – or saved in a file on a flop disk. Typically, in many interactive applications involving, say, multime pages comprising text and a passage of music, a synthesizer is first used create the passage of music which is then saved in a file. The author of t pages then links the file contents to the text at the point where the music is be played. Clearly, since the music is in the form of a sequence of MIDI m sages, it is necessary to have a **sound card** in the client computer to interpr the sequence of messages and generate the appropriate sounds. The soun generators use either **FM synthesis** techniques or samples of sound produce by real instruments. The latter is known as **wavelet synthesis**.

2.6 Video

Video features in a range of multimedia applications:

- entertainment: broadcast television and VCR/DVD recordings;
- interpersonal: video telephony and videoconferencing;
- interactive: windows containing short video clips.

The quality of the video required, however, varies considerably from on type of application to another. For example, for video telephony, a sma window on the screen of a PC is acceptable while for a movie, a large scree format is preferable. In practice, therefore, there is not just a single standar associated with video but rather a set of standards, each targeted at a particu lar application domain. Before describing a selection of these we must first acquire an understanding of the basic principles associated with broadcast television on which all the standards are based.

2.6.1 Broadcast television

We considered the basic principles of color television picture tubes earlier in Section 2.4.3 when the topic of digitized pictures was discussed. As you may recall, a color picture/image is produced from varying mixtures of the three primary colors red, green and blue. The screen of the picture tube is coated with a set of three different phosphors – one for each color – each of which is activated by a separate electron beam. The three electron beams are scanned in unison across the screen from left to right with a resolution of either 525 lines (NTSC) or 625 lines (PAL/CCIR/SECAM). The total screen contents are then refreshed at a rate of either 60 or 50 frames per second respectively, the rate being determined by the frequency of the mains electricity supply used in the different countries.

The computer monitors used with most personal computers use the same picture tubes as those in broadcast television receivers and hence operate in a similar way. The three digitized color signals that make up a stored picture/image are read from the computer memory in time-synchronism with the scanning operation of the display tube and, after each complete scan of the display, the procedure repeats so producing a flicker-free color image on the screen. In principle, broadcast television could operate in a similar way, but in practice it operates slightly differently both in terms of the scanning sequence used and in the choice of color signals. We shall look at each separately.

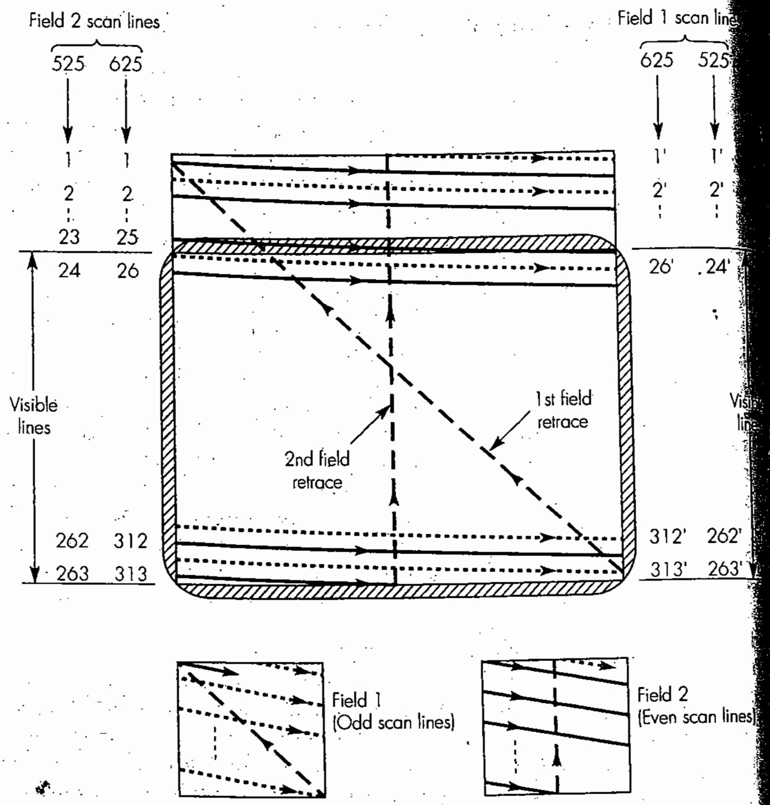
Scanning sequence

Although it is necessary to use a minimum refresh rate of 50 times per second to avoid flicker, to produce smooth motion, a refresh rate of 25 times per second is sufficient. Hence in order to minimize the amount of transmission bandwidth that is required to broadcast the television signal, this characteristic of the eye is exploited by transmitting the image/picture associated with each frame in two halves. Each is known as a **field**, the first comprising only the odd scan lines and the second the even scan lines. The two fields are then integrated together in the television receiver using a technique known as **interlaced scanning**, the principles of which are shown in Figure 2.19.

As we can see, in a 525-line system each field comprises 262.5 lines – 240 visible – while in a 625-line system each field comprises 312.5 lines – 288 visible, the remainder being used for other purposes. Each field is refreshed alternately at 60/50 fields per second and hence the resulting frame refresh rate is only 30/25 frames per second. As discussed, the higher field rate tricks the eye into thinking the frame rate is double what it is in practice. In this way, a refresh rate equivalent to 60/50 frames per second is achieved but with only half the transmission bandwidth.

Color signals

For historical reasons, the received signals associated with a color television broadcast had to be such that they could be used by an existing (unmodified) monochrome (black-and-white) television set to produce the same picture in high-quality monochrome. In addition, a color television had to be able to



525-line systems : 262.5 each field, 240 visible
625-line systems : 312.5 each field, 288 visible

Figure 2.19 Interlaced scanning principles.

produce black-and-white pictures from monochrome broadcasts. For the reasons a different set of color signals from R, G, and B were selected f color television broadcasts.

The three main properties of a color source that the eye makes u of are:

- **brightness:** this represents the amount of energy that stimulates the eye and varies on a gray scale from black (lowest) through to white (highest It is thus independent of the color of the source;
- **hue:** this represents the actual color of the source, each color has a differen frequency/wavelength and the eye determines the color from this;

- **saturation:** this represents the strength or vividness of the color, a pastel color has a lower level of saturation than a color such as red. Also, a saturated color such as red has no white light in it.

The term **luminance** is used to refer to the brightness of a source, and the hue and saturation, because they are concerned with its color, are referred to as its **chrominance** characteristics.

As we saw in Section 2.4.3, a range of colors can be produced by mixing the three primary colors R, G, and B. In a similar way, a range of colors can be produced on a television display screen by varying the magnitude of the three electrical signals that energize the red, green, and blue phosphors. For example, if the magnitude of the three signals are in the proportion

$$0.299R + 0.587G + 0.114B$$

then the color white is produced on the display screen. Hence, since the luminance of a source is only a function of the amount of white light it contains, for any color source its luminance can be determined by summing together the three primary components that make up the color in this proportion. That is,

$$Y_s = 0.299\ R_s + 0.587\ G_s + 0.144\ B_s$$

where Y_s is the amplitude of the luminance signal and R_s, G_s, and B_s are the magnitudes of the three color component signals that make up the source. Thus, since the luminance signal is a measure of the amount of white light it contains, it is the same as the signal used by a monochrome television. Two other signals, the **blue chrominance** (C_b), and the **red chrominance** (C_r), – are then used to represent the coloration – hue and saturation – of the source. These are obtained from the two **color difference** signals:

$$C_b = B_s - Y_s \quad \text{and} \quad C_r = R_s - Y_s$$

which, since the Y signal has been subtracted in both cases, contain no brightness information. Also, since Y is a function of all three colors, then G can be readily computed from these two signals. In this way, the combination of the three signals Y, C_b, and C_r contains all the information that is needed to describe a color signal while at the same time being compatible with monochrome televisions which use the luminance signal only.

Chrominance components

In practice, although all color television systems use this same basic principle to represent the coloration of a source, there are some small differences between the two systems in terms of the magnitude used for the two chrominance signals. These arise from the constraint that the bandwidth of the transmission channel for color broadcasts must be the same as that used for monochrome. As a result, in order to fit the Y, C_b, and C_r signals in the same bandwidth, the three signals must be combined together for transmission The resulting signal is then known as the **composite video signal.** As a result of doing this, however, if the two color difference signals are transmitted a their original magnitudes, the amplitude of the luminance signal can become greater than that of the equivalent monochrome signal. This leads to a degradation in the quality of the monochrome picture and hence is unacceptable.

To overcome this effect, the magnitude of the two color difference signals are both scaled down. In addition, since they both have different levels of luminance associated with them, the scaling factor used for each signal is different In practice, the two color difference signals are referred to by different symbols in each system. In the PAL system, for example, C_b and C_r are referred to as U and V respectively and the scaling factors used for the three signals are:

$$\text{PAL:}\quad \begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B \\ U &= 0.493\,(B - Y) \\ V &= 0.877\,(R - Y) \end{aligned}$$

In the case of the NTSC system, the two color difference signals are combined to form two different signals referred to as I and Q. The scaling factors used are:

$$\text{NTSC:}\quad \begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B \\ I &= 0.74\,(R - Y) - 0.27\,(B - Y) \\ Q &= 0.48\,(R - Y) + 0.41\,(B - Y) \end{aligned}$$

Example 2.6

Derive the scaling factors used for both the U and V (as used in PAL) and I and Q (as used in NTSC) color difference signals in terms of the three R, G, B color signals.

Answer:

PAL: $Y = 0.299\,R + 0.587\,G + 0.114\,B$

$U = 0.493\,(B - Y)$ and $V = 0.877\,(R - Y)$

Hence $U = 0.493\,B - 0.493\,(0.299\,R + 0.587\,G + 0.114\,B)$

$= -0.147\,R - 0.289\,G + 0.437\,B$

and $V = 0.877\,R - 0.877\,(0.299\,R + 0.587\,G + 0.114\,B)$

$= 0.615\,R - 0.515\,G - 0.100\,B$

NTSC: $I = 0.74\,(R - Y) - 0.27\,(B - Y)$

$= 0.74\,R - 0.27\,B - 0.47\,Y$

$= 0.599\,R - 0.276\,G - 0.324\,B$

$Q = 0.48\,(R - Y) + 0.41\,(B - Y)$

$= 0.48\,R + 0.41\,B - 0.89\,Y$

$= 0.212\,R - 0.528\,G + 0.311\,B$

Signal bandwidth

As we saw in the last section, the bandwidth of the transmission channel used for color broadcasts must be the same as that used for a monochrome broadcast. As a result, for transmission, the two chrominance signals must occupy the same bandwidth as that of the luminance signal. The baseband spectrum of a color television signal in both the NTSC and PAL systems is shown in parts (a) and (b) of Figure 2.20 respectively.

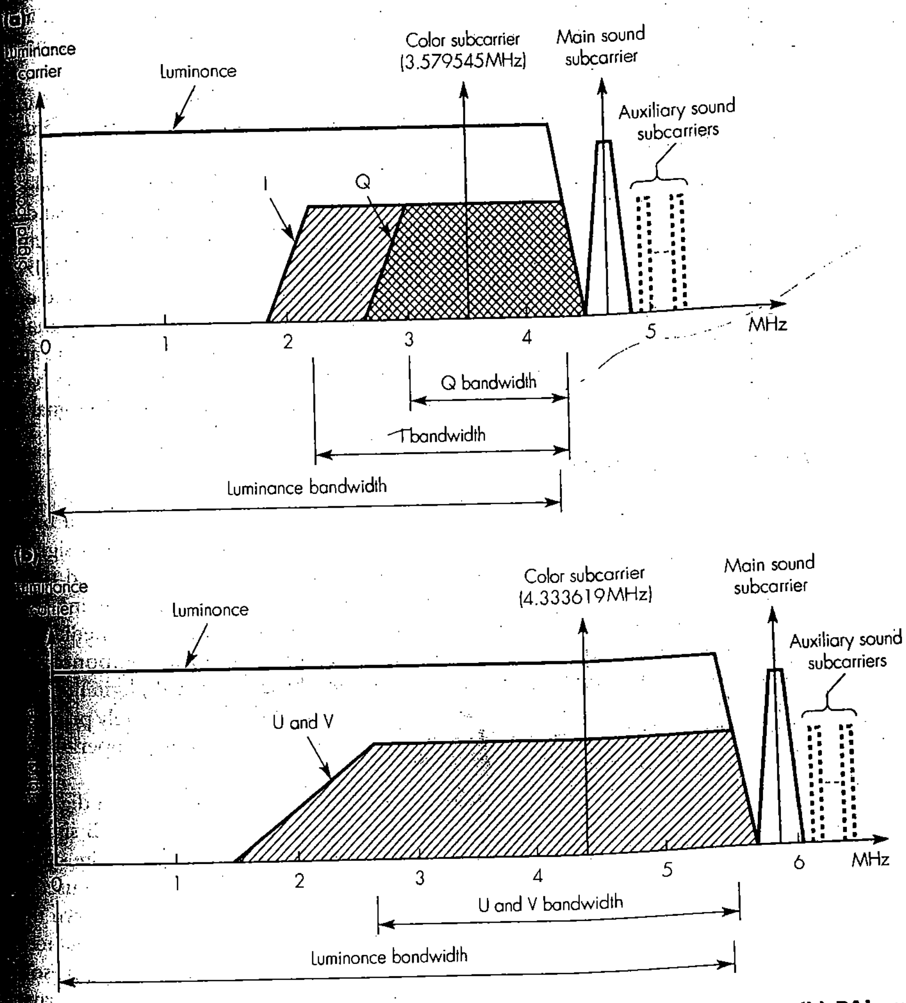

(a)
Luminance carrier
Signal power
Luminance
Color subcarrier (3.579545MHz)
Main sound subcarrier
Auxiliary sound subcarriers
I
Q
0
1
2
3
4
5
MHz
Q bandwidth
I bandwidth
Luminance bandwidth
(b)
Luminance carrier
Luminance
Color subcarrier (4.333619MHz)
Main sound subcarrier
Auxiliary sound subcarriers
U and V
0
1
2
3
4
5
6
MHz
U and V bandwidth
Luminance bandwidth

Figure 2.20 Baseband spectrum of color television signals: (a) NTSC system; (b) PAL system.

In practice, most of the energy associated with the luminance signal the lower frequency signals and hence the lower part of its frequency s trum. Hence in order to minimize the level of interference between luminance and two chrominance signals, firstly, the latter are transmitte the upper part of the luminance frequency spectrum using two separate carriers and secondly, to restrict the bandwidth used to the upper part of spectrum, a smaller bandwidth is used for both chrominance signals. In a tion, both of the two chrominance subcarriers have the same frequency they are 90 degrees out of phase with each other. Each is modulated indep dently in both amplitude and phase by the related chrominance signal. Us this technique, the two signals can use the same portion of the luminance quency spectrum.

In the NTSC system the eye is more responsive to the *I* signal than th signal. Hence to maximize the use of the available bandwidth while at the sa time minimizing the level of interference with the luminance signal, th signal has a modulated bandwidth of about 2 MHz and the *Q* signal a ba width of about 1 MHz. With the PAL system, the larger luminance bandwidt about 5.5 MHz relative to 4.2 MHz – allows both the *U* and *V* chrominan signals to have the same modulated bandwidth which is about 3 MHz. As show in the figure, the audio/sound signal is transmitted using one or mo separate subcarriers which are all just outside of the luminance signal ban width. Typically, the main audio subcarrier is for mono sound and the auxilia subcarriers are used for stereo sound. When these are added to the baseba video signal, the composite signal is called the **complex baseband signal**.

2.6.2 Digital video

In the previous section we described the underlying principles of broadca television and, in particular, the origin of the three component signals tha are used. In most multimedia applications the video signals need to be in digital form since it then becomes possible to store them in the memory o computer and to readily edit and integrate them with other media types. addition, although for transmission reasons the three component signa have to be combined for analog television broadcasts, with digital television is more usual to digitize the three component signals separately prior to the transmission. Again, this is done to enable editing and other operations to readily performed.

Since the three component signals are treated separately in digital tele sion, in principle it is possible simply to digitize the three RGB signals tha make up the picture. The disadvantage of this approach is that the same reso lution – in terms of sampling rate and bits per sample – must be used for al three signals. Studies on the visual perception of the eye have shown that th resolution of the eye is less sensitive for color than it is for luminance. Thi means that the two chrominance signals can tolerate a reduced resolution relative to that used for the luminance signal. Hence a significant saving in terms of the resulting bit rate – and hence transmission bandwidth – can be achieved by using the luminance and two color difference signals instead of the RGB signals directly.

Digitization of video signals has been carried out in television studios for many years in order, for example, to perform conversions from one video format into another. In order to standardize this process – and hence make the exchange of television programmes internationally easier – the international body for television standards, the **International Telecommunications Union – Radiocommunications Branch** (**ITU-R**) – formerly known as the **Consultative Committee for International Radiocommunications** (**CCIR**) – defined a standard for the digitization of video pictures known as **Recommendation CCIR-601**. In addition, a number of variants of this standard have been defined for use in other application domains such as digital television broadcasting, video telephony, and videoconferencing. Collectively these are known as **digitization formats** and, in practice, they all exploit the fact that the two chrominance signals can tolerate a reduced resolution relative to that used for the luminance signal. We shall now describe a selection of these.

4:2:2 format

This is the original digitization format used in Recommendation CCIR-601 for use in television studios. The three component (analog) video signals from a source in the studio can have bandwidths of up to 6 MHz for the luminance signal and less than half this for the two chrominance signals. To digitize these signals, as we described in Section 2.2, it is necessary to use band-limiting filters of 6 MHz for the luminance signal and 3 MHz for the two chrominance signals with a minimum sampling rate of 12 MHz (12 Msps) and 6 MHz respectively.

In the standard, however, a line sampling rate of 13.5 MHz for luminance and 6.75 MHz for the two chrominance signals was selected, both of which are independent of the particular scanning standard – NTSC, PAL and so on – being used. The 13.5 MHz rate was chosen since it is the nearest frequency to 12 MHz which results in a whole number of samples per line for both 525- and 625-line systems. The number of samples per line chosen is 702 and can be derived as follows:

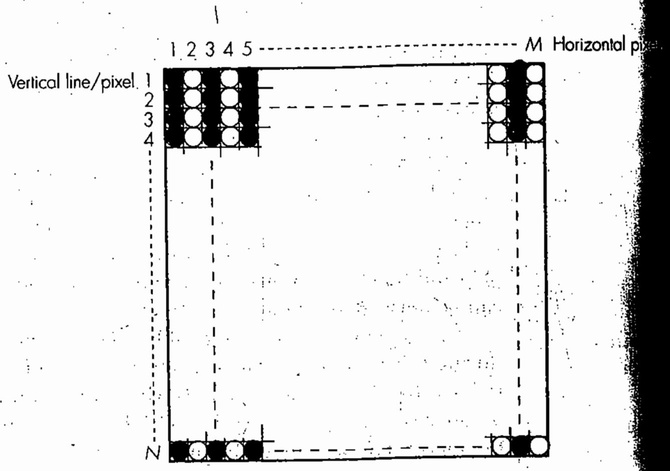
In a 525-line system, the total line sweep time is 63.56 microseconds but, during this time, the beam(s) is (are) turned off – set to the black level – for retrace for 11.56 microseconds which yields an active line sweep time of 52 microseconds. Similarly, in a 625-line system, the total line sweep time is 64 microseconds with a blanking time of 12 microseconds which also yields an active line sweep time of 52 microseconds. Hence in both cases, a sampling rate of 13.5 MHz yields:

$$52 \times 10^{-6} \times 13.5 \times 10^{6} = 702 \text{ samples per line}$$

In practice, the number of samples per line is increased to 720 by taking a slightly longer active line time which results in a small number of black samples at the beginning and end of each line for reference purposes. The corr ing number of samples for each of the two chrominance signals is the half this value; that is, 360 samples per active line. This results in 4*Y* sam every $2C_b$ and $2C_r$ samples which is the origin of the term 4:2:2, the te normally indicating the digitization is based on the R, G, B signals.

The number of bits per sample was chosen to be 8 for all three which corresponds to 256 quantization intervals. In addition, the vert olution for all three signals was also chosen to be the same, the number being determined by the scanning system in use; that is, 4 with a 525-line system and 576 lines with a 625-line system, the numb and 576 being the number of active (visible) lines in the respective Also, since the 4:2:2 format is intended for use in television studio interlaced scanning is used at a frame refresh rate of either 60 H 525-line system or 50 Hz for a 625-line system.

Since each line is sampled at a constant rate (13.5 and 6.75 MHz) fixed number of samples per line (720 and 360), the samples for ea are in a fixed position which repeats from frame to frame. The samp then said to be **orthogonal** and the sample method **orthogonal sam** Since each system (525 and 625) has a fixed number of (active) lin frame, the sampling positions for each of the three signals relative to a gular grid are as shown in Figure 2.21.



O = Y, + = C_b, X = C_r sample positions

525-line systems: $M = 720$, $N = 480$, 60 Hz refresh rate (interlace
$Y = 720 \times 480$, $C_b = C_r = 360 \times 480$

625-line systems: $M = 720$, $N = 576$, 50 Hz refresh rate (interlace
$Y = 720 \times 576$, $C_b = C_r = 360 \times 576$

Figure 2.21 Sample positions with 4:2:2 digitization format.

2.7

Derive the bit rate and the memory requirements to store each frame that result from the digitization of both a 525-line and a 625-line system assuming a 4:2:2 format. Also find the total memory required to store a 1.5 hour movie/video.

Answer:

525-line system: The number of samples per line is 720 and the number of visible lines is 480. Hence the resolution of the luminance (*Y*) and two chrominance (C_b and C_r) signals are:

$$Y = 720 \times 480$$
$$C_b = C_r = 360 \times 480$$

Bit rate: Line sampling rate is fixed at 13.5 MHz for *Y* and 6.75 MHz for both C_b and C_r, all with 8 bits per sample.

Hence: Bit rate $= 13.5 \times 10^6 \times 8 + 2\,(6.75 \times 10^6 \times 8) = 216$ Mbps

Memory required: Memory required per line $= 720 \times 8 + 2\,(360 \times 8)$
$= 11520$ bits or 1440 bytes

Hence memory per frame, each of 480 lines $= 480 \times 11520$
$= 5.5296$ Mbits or 691.2 kbytes

and memory to store 1.5 hours assuming 60 frames per second:
$= 691.2 \times 60 \times 1.5 \times 3600$ kbytes
$= 223.9488$ Gbytes

625-line system: Resolution: $Y = 720 \times 576$
$C_b = C_r = 360 \times 576$

Bit rate $= 13.5 \times 10^6 \times 8 + 2\,(6.75 \times 10^6 \times 8) = 216$ Mbps

Memory per frame $= 576 \times 11520 = 6.63555$ Mbits or 829.44 kbytes

and memory to store 1.5 hours assuming 50 frames per second:
$= 829.44 \times 50 \times 1.5 \times 3600$ kbytes
$= 223.9488$ Gbytes

It should be noted that in practice, the bit rate figures are less than the computed values since they include samples during the retrace time when the beam is switched off. Nevertheless, as we can deduce from the computed values, both the bit rate and the memory requirements are very large for both systems and it is for this reason that the various lower resolution formats have been defined.

4:2:0 format

This format is a derivative of the 4:2:2 format and is used in digital video broadcast applications. It has been found to give good picture quality and is derived by using the same set of chrominance samples for two consecutive lines. Since it is intended for broadcast applications, interlaced scanning is used and the absence of chrominance samples in alternative lines is the origin of the term 4:2:0. The position of the three sample instants per frame are as shown in Figure 2.22.

As we can see from the figure, this yields the same luminance resolution as the 4:2:2 format but half the chrominance resolution:

$$\begin{aligned} \text{525-line system:} \quad & Y = 720 \times 480 \\ & C_b = C_r = 360 \times 240 \end{aligned}$$

$$\begin{aligned} \text{625-line system:} \quad & Y = 720 \times 576 \\ & C_b = C_r = 360 \times 288 \end{aligned}$$

The bit rate in both systems with this format is:

$$13.5 \times 10^6 \times 8 + 2\,(3.375 \times 10^6 \times 8) = 162 \text{ Mbps}$$

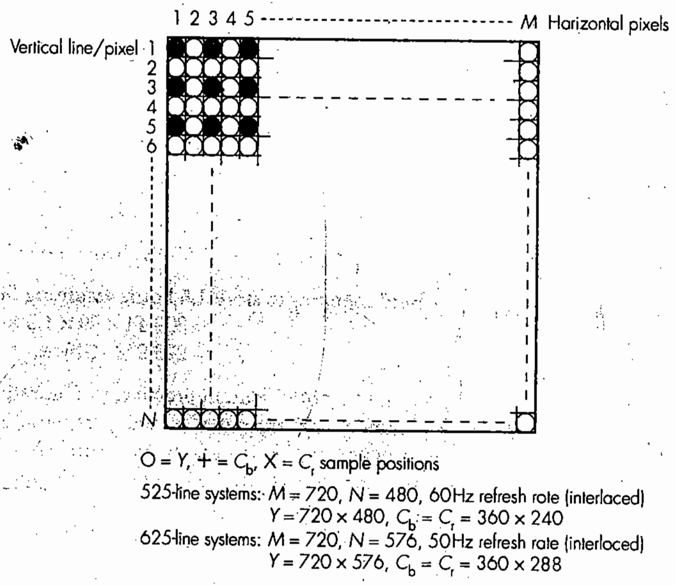

1 2 3 4 5
M Harizontal pixels
Vertical line/pixel 1
2
3
4
5
6
N
O = Y, + = C_b, X = C_r sample positions
525-line systems: M = 720, N = 480, 60Hz refresh rote (interlaced)
Y = 720 × 480, C_b = C_r = 360 × 240
625-line systems: M = 720, N = 576, 50Hz refresh rate (interloced)
Y = 720 × 576, C_b = C_r = 360 × 288

Figure 2.22 Sample positions in 4:2:0 digitization format.

It should be noted, however, that, as we pointed out in Example 2.7, this is the worst-case bit rate since it includes samples during the retrace times when the beam is switched off. Also, to avoid flicker effects with the chrominance signals, the receiver uses the same chrominance values from the sampled lines for the missing lines. With large-screen televisions, flicker effects are often reduced further by the receiver storing the incoming digitized signals of each field in a memory buffer. A refresh rate of double the normal rate – 100/120 Hz – is then used with the stored set of signals used for the second field.

HDTV formats

There are a number of alternative digitization formats associated with high-definition television (HDTV). The resolution of those which relate to the older 4/3 aspect ratio tubes can be up to 1440 × 1152 pixels and the resolution of those which relate to the newer 16/9 wide-screen tubes can be up to 1920 × 1152 pixels. In both cases, the number of visible lines per frame is 1080 which produces a square-pixel lattice structure with both tube types. Both use either the 4:2:2 digitization format for studio applications or the 4:2:0 format for broadcast applications. The corresponding frame refresh rate is either 50/60 Hz with the 4:2:2 format or 25/30 Hz with the 4:2:0 format. Hence in the case of the 1440 × 1152 resolution, the resulting worst-case bit rates are four times the values derived in the previous two sections and proportionally higher for the wide-screen format.

SIF

The **source intermediate format** (SIF) has been found to give a picture quality comparable with that obtained with video cassette recorders (VCRs). It uses half the spatial resolution in both horizontal and vertical directions as that used in the 4:2:0 format – a technique known as **subsampling** – and, in addition, uses half the refresh rate – also known as **temporal resolution**. This means that the frame refresh rate is 30 Hz for a 525-line system and 25 Hz for a 625-line system. Since the SIF is intended for storage applications, progressive (non-interlaced) scanning is used. The digitization format is known, therefore, as 4:1:1. The positions of the three sampling instants per frame are as shown in Figure 2.23.

As we can deduce from the figure, this yields resolutions of:

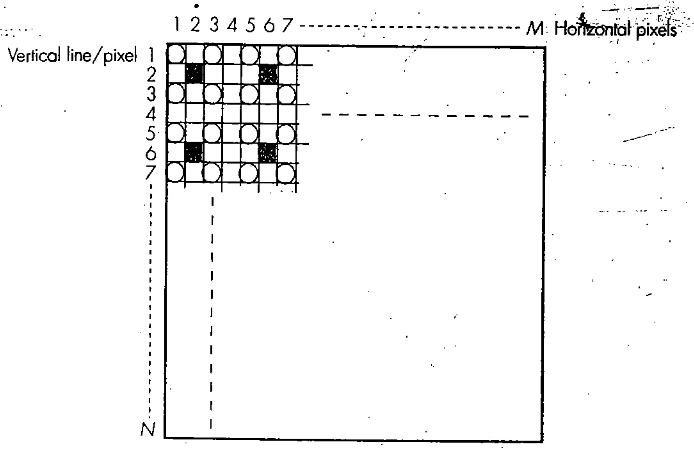
$$\begin{aligned} \text{525-line system:} \quad & Y = 360 \times 240 \\ & C_b = C_r = 180 \times 120 \end{aligned}$$

$$\begin{aligned} \text{625-line system:} \quad & Y = 360 \times 288 \\ & C_b = C_r = 180 \times 144 \end{aligned}$$

The worst-case bit rate in both systems with this format is:

$$6.75 \times 10^6 \times 8 + 2(1.6875 \times 10^6 \times 8) = 81 \text{ Mbps}$$

At the receiver, the missing samples are estimated by interpolating between each pair of values that are sent.



O = Y, + = C_b, X = C_r sample positions

SIF: 525-line systems: $M = 720$, $N = 480$ with 30 Hz refresh rate (non-interlaced)
$Y = 360 \times 240$, $C_b = C_r = 180 \times 120$
625-line systems: $M = 720$, $N = 576$ with 25 Hz refresh rate (non-interlaced)
$Y = 360 \times 288$, $C_b = C_r = 180 \times 144$
CIF: $M = 720$, $N = 576$ with 30 Hz refresh rate (non-interlaced)
$Y = 360 \times 288$, $C_b = C_r = 180 \times 144$

Figure 2.23 Sample positions for SIF and CIF.

CIF

The **common intermediate format (CIF)** has been defined for use in videoconferencing applications. It is derived from the SIF and uses a combination of the spatial resolution used for the SIF in the 625-line system and the temporal resolution used in the 525-line system. This yields a spatial resolution of:

$$Y = 360 \times 288$$
$$C_b = C_r = 180 \times 144$$

with a temporal resolution of 30 Hz using progressive scanning. The positions of the sampling instants per frame are the same as for SIF and hence the digitization format is 4:1:1. Similarly, the worst-case bit rate is the same and hence is 81 Mbps. As we can deduce from this, to convert to the CIF, a 525-line system needs a line-rate conversion and a 625-line system a frame-rate conversion.

In addition to the basic CIF, a number of higher-resolution derivatives of the CIF have been defined. As we described earlier in Section 1.4.1, there are a number of different types of videoconferencing applications including those that involve a linked set of desktop PCs and those that involve a linked set of videoconferencing studios. In general, therefore, because most desktop applications use switched circuits, a typical bit rate used is a single 64 kbps ISDN channel. For linking videoconferencing studios, however, dedicated circuits are normally used that comprise multiple 64 kbps channels. Hence because the bit rate of these circuits is much higher – typically four or sixteen 64 kbps channels – then a higher-resolution version of the basic CIF can be used to improve the quality of the video. Two examples are:

$$\text{4CIF:} \quad Y = 720 \times 576$$
$$C_b = C_r = 360 \times 288$$
$$\text{16CIF:} \quad Y = 1440 \times 1152$$
$$C_b = C_r = 720 \times 576$$

QCIF

The **quarter CIF (QCIF)** format has been defined for use in video telephony applications. It is derived from the CIF and uses half the spatial resolution of CIF in both horizontal and vertical directions and the temporal resolution is divided by either 2 or 4. This yields a spatial resolution of:

$$Y = 180 \times 144$$
$$C_b = C_r = 90 \times 72$$

with a temporal resolution of either 15 or 7.5 Hz. The worst-case bit rate with this format is:

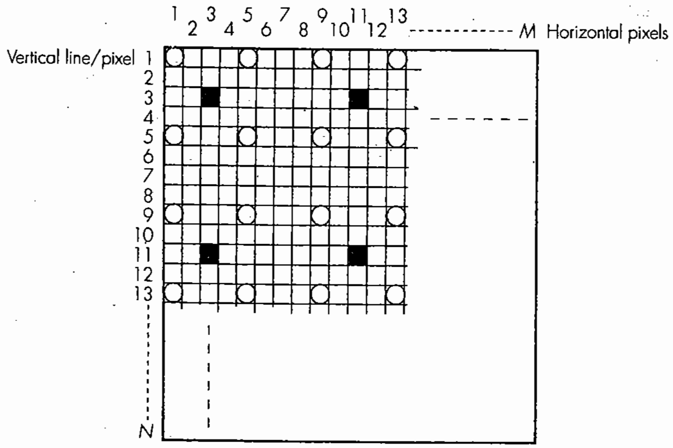
$$3.375 \times 10^6 \times 8 + 2\ (0.84375 \times 10^6 \times 8) = 40.5\,\text{Mbps}$$

The positions of the three sampling instants per frame are as shown in Figure 2.24 and, as we can see, it has the same 4:1:1 digitization format as CIF.

As we described in Section 1.4.1, a typical video telephony application involves a single switched 64 kbps channel and the QCIF is intended for use with such channels. In addition, there are lower-resolution versions of the QCIF which are intended for use in applications that use lower bit rate channels such as that provided by a modem and the PSTN. These lower-resolution versions are known as **sub-QCIF** or **S-QCIF** and an example is:

$$Y = 128 \times 96$$
$$C_b = C_r = 64 \times 48$$

It should be noted, however, that although the sampling matrix appears sparse, in practice, only a small screen (or a small window of a larger screen) is normally used for video telephony and hence the total set of samples may occupy all the pixel positions on the screen or window.



O = Y, + = C_b, X = C_r sample positions
$M = 720$, $N = 576$ with 15 Hz or 7.5 Hz refresh rate (non-interlaced)
$Y = 180 \times 144$, $C_b = C_r = 90 \times 72$

Figure 2.24 Sample positions for QCIF.

2.6.3 PC video

All the digitization formats described in Section 2.6.2 are intended for use with standard television receivers. However, as we discussed in Chapter 1, a number of multimedia applications that involve live video, use a window on the screen of a PC monitor for display purposes. Examples include desktop video telephony and videoconferencing, and also **video-in-a-window**.

As we discussed under the subheading "Aspect ratio" in Section 2.4.3, in order to avoid distortion on a PC screen – for example when displaying a square of $N \times N$ pixels – it is necessary to use a horizontal resolution of 640 (480 × 4/3) pixels per line with a 525-line PC monitor and 768 (576 × 4/3) pixels per line with a 625-line PC monitor. Hence for multimedia applications that involve mixing live video with other information on a PC screen, the line sampling rate is normally modified in order to obtain the required horizontal resolution.

To achieve the necessary resolution with a 525-line monitor, the line sampling rate is reduced from 13.5 MHz to 12.2727 MHz while for a 625-line monitor, the line sampling rate must be increased from 13.5 MHz to 14.75 MHz. In the case of desktop video telephony and videoconferencing, the video signals from the camera are sampled at this rate prior to transmission and hence can be displayed directly on a PC screen. In the case of a digital television broadcast a conversion is necessary before the video is displayed. The various digitization formats for use with PC video are as shown in Table 2.2. It should be remembered that all PC monitors use progressive (non-interlaced) scanning rather than interlaced scanning.

Table 2.2 PC video digitization formats.

Digitization format	System	Spatial resolution	Temporal resolution
[illegible]	525-line	$Y = 640 \times 480$ $C_b = C_r = 320 \times 240$	60 Hz
	625-line	$Y = 768 \times 576$ $C_b = C_r = 384 \times 288$	50 Hz
[illegible]	525-line	$Y = 320 \times 240$ $C_b = C_r = 160 \times 240$	30 Hz
	625-line	$Y = 384 \times 288$ $C_b = C_r = 192 \times 144$	25 Hz
[illegible]		$Y = 384 \times 288$ $C_b = C_r = 192 \times 144$	30 Hz
[illegible]		$Y = 192 \times 144$ $C_b = C_r = 96 \times 72$	15/7.5 Hz

2.6.4 Video content

In the preceding section we described the various digitization formats that have been defined for use in different application domains of digital video. In terms of the actual video content, therefore, this depends on the particular application. For example, in entertainment applications, the content will be either a broadcast television program or, in a video-on-demand application, a digitized movie that is accessed from a suitable server. Similarly, in interpersonal applications such as video telephony and videoconferencing, the video source will be derived from a video camera and the digitized sequence of pixels relating to each frame are transmitted across the network. As the pixels are received at the destination, they are displayed directly on either a television screen or a computer monitor.

In addition, in many interactive applications that involve video, the short video clips associated with the application are obtained by plugging a video camera into a **video capture board** within the computer that is preparing the (interactive) page contents. Normally, the computer stores the digitized video produced by such boards into a file ready for linking to the other page contents.

In other applications the video may be generated by a computer program rather than a video camera. This type of video content is normally referred to as **computer animation** or sometimes, because of the way it is generated,

animated graphics. A range of special programming languages is available f creating computer animation. Hence in the same way that a graphical imag produced by a graphics program can be represented in the form of either high-level program or a pixel image, so a computer animation can be repr sented in the form of either an animation program or a digital video. A before, the form used depends on the application. In general the digit video form of representation of an animation requires considerably mor memory and transmission bandwidth than the corresponding high-level pro gram form.

The negative side of a high-level program form is that the low-level ani mation primitives that the program uses – move object, rotate object, objec fill, and so on – have to be executed very fast in order to produce smooth motion on the display. Hence it is now common to have an additional (3-D) **graphics accelerator** processor to carry out these functions. Typically, the (host) processor simply passes the sequence of low-level primitives to the accelerator processor at the appropriate rate. The accelerator then, in turn executes each set of primitives to produce the corresponding pixel image in the video RAM at the desired refresh rate.

Summary

In this chapter we have described the different ways that the range of media types associated with the various multimedia applications we identified in Chapter 1 are represented in their digital form. These included various types of text, images, digitized documents and pictures, audio – both speech and music – and video. In the case of audio and video, the conversion operations that are used to convert them from their source analog form into their corresponding digital form were also described. In practice, with these basic forms of representation, the amount of bandwidth that is required to transfer the total quantity of information associated with a particular application is considerably larger than that which is available with many of the communication networks used for these applications.

For example, the bandwidth available for digital television – in terms of bits per second – is in the order of 40 Mbps with cable and terrestrial broadcast systems and 60 Mbps with a satellite channel. Clearly, these are both still considerably less than the bit rate requirement of 162 Mbps that is generated using the 4:2:0 digitization format. Similarly, the bit rate available with a connection through the all-digital ISDN is between 64 kbps and 2 Mbps and hence again the bandwidth requirements for videoconferencing – 81 Mbps with the CIF – and video telephony – 40.5 Mbps with the QCIF – are both far in excess of these two values. Hence in order to provide such services over the related networks, it is necessary to reduce the bandwidth requirements of the source signals considerably. In addition, when using public networks such as a PSTN or an ISDN in which call charges are based on the duration of a call, considerable cost savings can be made if the amount of data to be transmitted is reduced. For example, if the amount of data is reduced by a factor of two, then the cost of the call will be halved.

In most multimedia applications, therefore, a technique known as compression is first applied to the source information prior to its transmission. This is done either to reduce the volume of data to be transmitted – for example with text, fax, and images – or to reduce the bandwidth that is required for its transmission – for example with speech, audio, and video. In the next chapter we describe a range of compression algorithms that have been developed to reduce the volume of the data associated with text, fax, and images and, in Chapter 4, a range of compression algorithms associated with audio and video.

Exercises

Section 2.1

2.1 Explain the meaning of the following terms:
  (i) codeword,
  (ii) analog signal,
  (iii) signal encoder,
  (iv) signal decoder.

Section 2.2

2.2 With the aid of a set of signal waveforms, show the principles of how a time-varying analog signal is made up of a range of sinusoidal frequency components of differing amplitude and phase relative to one another.

2.3 Define the term "signal bandwidth". Hence show in graphical form the bandwidth of a speech signal and a music signal. Clearly show the dimensions of the horizontal and vertical axes.

2.4 Define the meaning of the term "channel bandwidth" in relation to a transmission channel. Hence with the aid of a diagram, explain the meaning of the term "bandlimiting channel".

2.5 Use a diagram to identify the main circuit components associated with a signal encoder. Hence by means of an associated set of signal waveforms, explain the meaning of the terms:
  (i) bandlimiting filter,
  (ii) ADC,
  (iii) sample-and-hold,
  (iv) quantizer.

2.6 Explain the meaning of the following terms relating to the sampling of an analog signal:
  (i) Nyquist sampling theorem,
  (ii) Nyquist rate.

2.7 Show by means of a diagram how sampling of an analog signal at a rate lower than the Nyquist rate can generate additional lower-frequency alias signals to those present in the original waveform. How can this be avoided?

2.8 Define the meaning of the term "quantization interval" and how this influences the accuracy of the sampling process of an analog signal. Hence with the aid of a diagram, explain the meaning of the terms "quantization error" and "quantization noise".

2.9 State the meaning of the term "dynamic range" as applied to an analog signal and show how this is expressed in decibels. How does this influence the number of bits to be used for the quantizer part of an ADC?

2.10 With the aid of a diagram and an associated waveform set, explain the function of the following components that make up a signal decoder:
  (i) DAC,
  (ii) low-pass filter.

Why is the latter also known as a recovery/reconstruction filter?

on 2.3

State the meaning of the following types of text:
(i) unformatted/plaintext,
(ii) formatted/richtext,
(iii) hypertext.

By means of examples, show how the 7-bit ASCII character set can be extended to create additional characters and symbols. State one of the uses of an extended character set.

How is formatted text different from unformatted text? Hence describe the meaning of the term "text" and "document formatting commands". What is the origin of the acronym WYSIWYG?

Describe the terms "hypertext", "pages/documents", and "hyperlinks".

With the aid of diagrams where appropriate, describe the meaning of the following terms relating to HTML and the World Wide Web:
(i) browser,
(ii) home page,
(iii) URL,
(iv) page formatting commands.

on 2.4

Explain the meaning of the following terms relating to graphical images:
(i) visual object,
(ii) freeform object,
(iii) clip-art,
(iv) 3-D objects.

With the aid of diagrams, explain the meaning of the following terms relating to graphical images:
(i) pixels,
(ii) video graphics array,
(iii) image object,
(iv) object attributes,
(v) open and closed object shapes,
(vi) rendering,
(vii) bit-map format.

2.18 With the aid of a diagram, explain the me ing of the following terms relating to facsim machines:
(i) scanning,
(ii) pels,
(iii) digitization resolution.

2.19 What is the difference between a bitonal imag and a continuous-tone image?

2.20 With the aid of diagrams where appropria explain the meaning of the terms:
(i) color gamut,
(ii) additive color mixing,
(iii) subtractive color mixing.

Give an application of both color mixin methods.

2.21 With the aid of diagrams, describe the raste scan operation associated with TV/compute monitors. Include in your description th meaning of the terms:
(i) line scan,
(ii) horizontal and vertical retrace,
(iii) phosphor triad,
(iv) frame refresh rate,
(v) flicker,
(vi) pixel depth,
(vii) video RAM,
(viii) video controller.

2.22 Define the aspect ratio of a display screen Give two examples for current widely used screen sizes.

2.23 What is the number of scan lines per frame asso ciated with each of the following TV monitors:
(i) NTSC,
(ii) PAL?

In practice, the number of visible lines pe frame are less than these values. State wha these are for each type of monitor. In each case, derive the number of pixels per scan line that are used to obtain square pixels assuming a 4/3 aspect ratio.

2.24 Most high resolution computer monitors are not based on television picture tubes. What is the amount of memory that is required to

an image with each of the following display sizes:
(i) 1024 × 768,
(ii) 1280 × 1024?

Derive the time to transmit an image with each type of display assuming a bit rate of
(i) 56 kbps,
(ii) 1.5 Mbps.

2.25 With the aid of a diagram, explain how a digital image produced by a scanner or digital camera is captured and stored within the memory of a computer.

2.26 With the aid of a diagram, explain how a color image is captured within a camera or scanner using each of the following methods:
(i) single image sensor,
(ii) a single image sensor with filters,
(iii) three separate image sensors. Include in your explanations the terms "photosites" and "CCDs" and the role of the readout register.

Section 2.5

2.27 With the aid of a diagram, explain the principle of operation of a PCM speech codec. Include in your diagram the operation of the compressor in the encoder and the expander in the decoder. Use for example purposes 5 bits per sample.

Identify the main features of the MIDI standard and its associated messages.

on 2.6

With the aid of a diagram, explain the principles of interlaced of scanning as used in most TV broadcast applications. Include in your explanation the meaning of the terms "field", "odd scan lines", and "even scan lines". Show the number of scan lines per field with
(i) a 525-line system and
(ii) a 625-line system. Why do computer monitors not use interlaced scanning?

2.30 State and explain the three main properties of a color source that the eye makes of. Hence explain the meaning of the terms "luminance", "chrominance", and "color difference" and how the magnitude of each primary color present in the source is derived from these.

2.31 Why is the chrominance signal transmitted in the form of two color different signals? Identify the color difference signals associated with the NTSC and PAL systems.

2.32 State the meaning of the term "composite video signal" and, with the aid of a diagram, describe how the two color difference signals are transmitted within the same frequency band as that used for the luminance signal.

2.33 Explain why, for digital TV transmission, the three digitized signals used are the luminance and two color difference signals rather than the RGB signals. Why are a number of different digitization formats used?

2.34 With the aid of diagrams, describe the following digitization formats:
(i) 4:2:2,
(ii) 4:2:0,
(iii) SIF,
(iv) CIF,
(v) QCIF,
(vi) S-QCIF.

For each format, state the temporal resolution and the sampling rate used for the luminance and the two color difference signals. Give an example application of each format.

2.35 Derive the bit rate that results from the digitization of a 525-line and a 625-line system using the 4:2:0 digitization format and interlaced scanning. Hence derive the amount of memory required to store a 2-hour movie/ video.

2.36 Explain why modifications to the received (broadcast) TV signal have to be made if the signal is to be displayed in a window of a computer monitor. Hence assuming the SIF format, derive the spatial resolution required with
(i) a 525-line and
(ii) a 625-line system.

3

Text and image compression

3.1 Introduction

In the previous chapter we described the way the different types of used in multimedia applications – text, fax, images, speech, audio, and – are represented in a digital form. We derived the memory and ban requirements for each type and, as we concluded in Section 2.7, i cases, the bandwidths derived were greater than those that are availab the communication networks over which the related services are provi addition, when using a public network in which call charges are based duration of a call, considerable cost savings can be made if the vol information to be transmitted is reduced.

In almost all multimedia applications, therefore, a technique kno **compression** is first applied to the source information prior to its tra sion. This is done either to reduce the volume of information transmitted – text, fax, and images – or to reduce the bandwidth required for its transmission – speech, audio, and video. In this chapt shall consider a selection of the compression algorithms which are used text, fax, and images and, in Chapter 4, we shall describe a selection o compression algorithms that are used with audio and video.

3.2 Compression principles

Before we describe some of the compression algorithms in widespread use, it will be helpful if we first build up an understanding of the principles on which they are based. We shall discuss the under the headings:

- source encoders and destination decoders,
- lossless and lossy compression,
- entropy encoding,
- source encoding.

3.2.1 Source encoders and destination decoders

As have just indicated, prior to transmitting the source information relating to a particular multimedia application, a compression algorithm is applied to it. This implies that in order for the destination to reproduce the original source information – or, in some instances, a nearly exact copy of it – a matching decompression algorithm must be applied to it. The application of the compression algorithm is the main function carried out by the **source encoder** and the decompression algorithm is carried out by the **destination decoder.**

In applications which involve two computers communicating with each other, the time required to perform the compression and decompression algorithms is not always critical. So both algorithms are normally implemented in software within the two computers. The general scheme is shown in part (a) of Figure 3.1 and an example application which uses this approach is the compression of text and/or image files. In other applications, however, the time required to perform the compression and decompression algorithms in software is not acceptable and instead the two algorithms must be performed by special processors in separate units as shown in part (b) of the figure. Example applications which use this approach are those which involve speech, audio, and video.

3.2.2 Lossless and lossy compression

Compression algorithms can be classified as being either lossless or lossy. In the case of a **lossless compression algorithm** the aim is to reduce the amount of source information to be transmitted in such a way that, when the compressed information is decompressed, there is no loss of information. Lossless compression is said, therefore, to be **reversible.** An example application of lossless compression is for the transfer over a network of a text file since, in such applications, it is normally imperative that no part of the source information is lost during either the compression or decompression operations.

In contrast, the aim of **lossy compression algorithms,** is normally not to reproduce an exact copy of the source information after decompression but

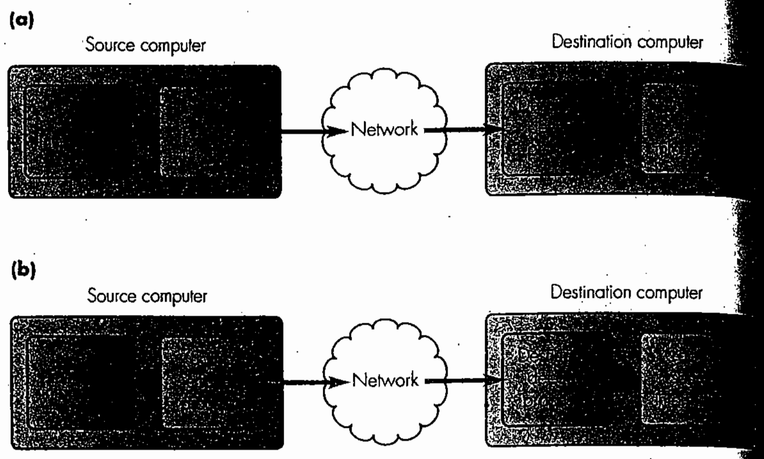


Figure 3.1 Source encoder/destination decoder alternatives: (a) software only; (b) special processors/hardware.

rather a version of it which is perceived by the recipient as a true copy. In ge eral, with such algorithms the higher the level of compression being appli to the source information the more approximate the received versio becomes. Example applications of lossy compression are for the transfer digitized images and audio and video streams. In such cases, the sensitivity the human eye or ear is such that any fine details that may be missing fro the original source signal after decompression are not detectable.

3.2.3 Entropy encoding

Entropy encoding is lossless and independent of the type of information th is being compressed. It is concerned solely with how the information is rep sented. We shall describe two examples which are in widespread use compression algorithms in order to illustrate the principles involved. In so applications they are used separately while in others they are used together

Run-length encoding

Typical applications of this type of encoding are when the source informatio comprises long substrings of the same character or binary digit. Instead o transmitting the source string in the form of independent codewords or bit it is transmitted in the form of a different set of codewords which indicate n only the particular character or bit being transmitted but also an indicatio of the number of characters/bits in the substring. Then, providing the dest nation knows the set of codewords being used, it simply interprets eac codeword received and outputs the appropriate number of characters or bits

For example, in an application that involves the transmission of long strings of binary bits that comprise a limited number of substrings, each substring can be assigned a separate codeword. The total bit string is then transmitted in the form of a string of codewords selected from the codeword set. An example application which uses this technique is for the transmission of the binary strings produced by the scanner in a facsimile machine. In many instances – for example when scanning typed documents – the scanner produces long substrings of either binary 0s or 1s. Instead of transmitting these directly, they are sent in the form of a string of codewords, each indicating both the bit – 0 or 1 – and the number of bits in the substring. For example, if the output of the scanner was:

000000011111111110000011...

this could be represented as: 0,7 1,10 0,5 1,2 ... . Alternatively, since only the two binary digits 0 and 1 are involved, if we ensure the first substring always comprises binary 0s, then the string could be represented as 7, 10, 5, 2 ... . To send this in a digital form, the individual decimal digits would be sent in their binary form and, assuming a fixed number of bits per codeword, the number of bits per codeword would be determined by the largest possible substring. We shall describe an application that uses this approach in Section 3.4.3 when we describe the compression of digitized documents.

Statistical encoding

Many applications use a set of codewords to transmit the source information. For example, as we described earlier in Section 2.3.1, a set of ASCII codewords are often used for the transmission of strings of characters. Normally, all the codewords in the set comprise a fixed number of binary bits, for example 7 bits in the case of ASCII. In many applications, however, the symbols – and hence codewords – that are present in the source information do not occur with the same frequency of occurrence; that is, with equal probability. For example, in a string of text, the character A may occur more frequently than, say, the character P which occurs more frequently than the character Z, and so on. Statistical encoding exploits this property by using a set of variable-length codewords with the shortest codewords used to represent the most frequently occurring symbols.

In practice, the use of variable-length codewords is not quite as straightforward as it first appears. Clearly, as with run-length encoding, the destination must know the set of codewords being used by the source. With variable-length codewords, however, in order for the decoding operation to be carried out correctly, it is necessary to ensure that a shorter codeword in the set does not form the start of a longer codeword otherwise the decoder will interpret the string on the wrong codeword boundaries. A codeword set that avoids this happening is said to possess the **prefix property** and an example of an encoding scheme that generates codewords that have this property is the **Huffman encoding algorithm** which we shall describe in Section 3.3.1.

The theoretical minimum average number of bits that are required to transmit a particular source stream is known as the **entropy** of the source and can be computed using a formula attributed to Shannon:

$$\text{Entropy, } H = -\sum_{i=1}^{n} P_i \log_2 P_i$$

where n is the number of different symbols in the source stream and P_i is the probability of occurrence of symbol i. Hence the efficiency of a particular encoding scheme is often computed as a ratio of the entropy of the source to the average number of bits per codeword that are required with the scheme. The latter is computed using the formula:

$$\text{Average number of bits per codeword} = \sum_{i=1}^{n} N_i P_i$$

ample 3.1

A statistical encoding algorithm is being considered for the transmission of a large number of long text files over a public network. Analysis of the file contents has shown that each file comprises only the six different characters M, F, Y, N, 0, and 1 each of which occurs with a relative frequency of occurrence of 0.25, 0.25, 0.125, 0.125, 0.125, and 0.125 respectively. If the encoding algorithm under consideration uses the following set of codewords:

M = 10, F = 11, Y = 010, N = 011, 0 = 000, 1 = 001

compute:

(i) the average number of bits per codeword with the algorithm,

(ii) the entropy of the source,

(iii) the minimum number of bits required assuming fixed-length codewords.

Answer:

(i) Average number of bits per codeword

$$= \sum_{i=1}^{6} N_i P_i = (2(2\times0.25) + 4(3\times0.125))$$

$$= 2\times0.5 + 4\times0.375 = 2.5$$

Continued

(ii) Entropy of source

$$= -\sum_{i=1}^{6} P_i \log_2 P_i = -(2(0.25\log_2 0.25) + 4(0.125\log_2 0.125))$$

$$= 1 + 1.5 = 2.5$$

(iii) Since there are 6 different characters, using fixed-length codewords would require a minimum of 3 bits (8 combinations).

3.2.4 Source encoding

Source encoding exploits a particular property of the source information in order to produce an alternative form of representation that is either a compressed version of the original form or is more amenable to the application of compression. Again, we shall describe two examples in widespread use in order to illustrate the principles involved.

Differential encoding

Differential encoding is used extensively in applications where the amplitude of a value or signal covers a large range but the difference in amplitude between successive values/signals is relatively small. To exploit this property of the source information, instead of using a set of relatively large codewords to represent the amplitude of each value/signal, a set of smaller codewords can be used each of which indicates only the difference in amplitude between the current value/signal being encoded and the immediately preceding value/signal. For example, if the digitization of an analog signal requires, say, 12 bits to obtain the required dynamic range but the maximum difference in amplitude between successive samples of the signal requires only 3 bits, then by using only the difference values a saving of 75% on transmission bandwidth can be obtained.

In practice, differential encoding can be either lossless or lossy and depends on the number of bits used to encode the difference values. If the number of bits used is sufficient to cater for the maximum difference value then it is lossless. If this is not the case, then on those occasions when the difference value exceeds the maximum number of bits being used, temporary loss of information will result.

Transform encoding

As the name implies, transform encoding involves transforming the source information from one form into another, the other form lending itself more readily to the application of compression. In general, there is no loss of information associated with the transformation operation and this technique is used in a number of applications involving both images and video. For example, as we saw in Section 2.4.3, the digitization of a continuous-tone monochromatic image produces a two-dimensional matrix of pixel values each of which represents the level of gray in a particular position of the image. As we go from one position in the matrix to the next, the magnitude of each pixel value may vary. Hence, as we scan across a set of pixel locations, the rate of change in magnitude will vary from zero, if all the pixel values remain the same, to a low rate of change if, say, one half is different from the next half, through to a high rate of change if each pixel magnitude changes from one location to the next. Some examples are shown in Figure 3.2(a).

The rate of change in magnitude as one traverses the matrix gives rise to a term known as **spatial frequency** and, for any particular image, there will be a mix of different spatial frequencies whose amplitudes are determined by the related changes in magnitude of the pixels. This is true, of course, if we scan the matrix in either the horizontal or the vertical direction and this, in turn, gives rise to the terms **horizontal** and **vertical frequency components** of the image. In practice, the human eye is less sensitive to the higher spatial frequency components associated with an image than the lower frequency components. Moreover, if the amplitude of the higher frequency components falls below a certain amplitude threshold, they will not be detected by the eye. Hence in terms of compression, if we can transform the original spatial form of representation into an equivalent representation involving spatial frequency components, then we can more readily identify and eliminate those higher frequency components which the eye cannot detect thereby reducing the volume of information to be transmitted without degrading the perceived quality of the original image.

The transformation of a two-dimensional matrix of pixel values into an equivalent matrix of spatial frequency components can be carried out using a mathematical technique known as the **discrete cosine transform (DCT)**. The transformation operation itself is lossless – apart from some small rounding errors in the mathematics – but, once the equivalent matrix of spatial frequency components – known as coefficients – has been derived, then any frequency components in the matrix whose amplitude is less than a defined threshold can be dropped. It is only at this point that the operation becomes lossy. The basic principle behind the DCT is as shown in Figure 3.2(b) and we shall describe it in more detail in Section 3.4.5 when we discuss the topic of image compression.

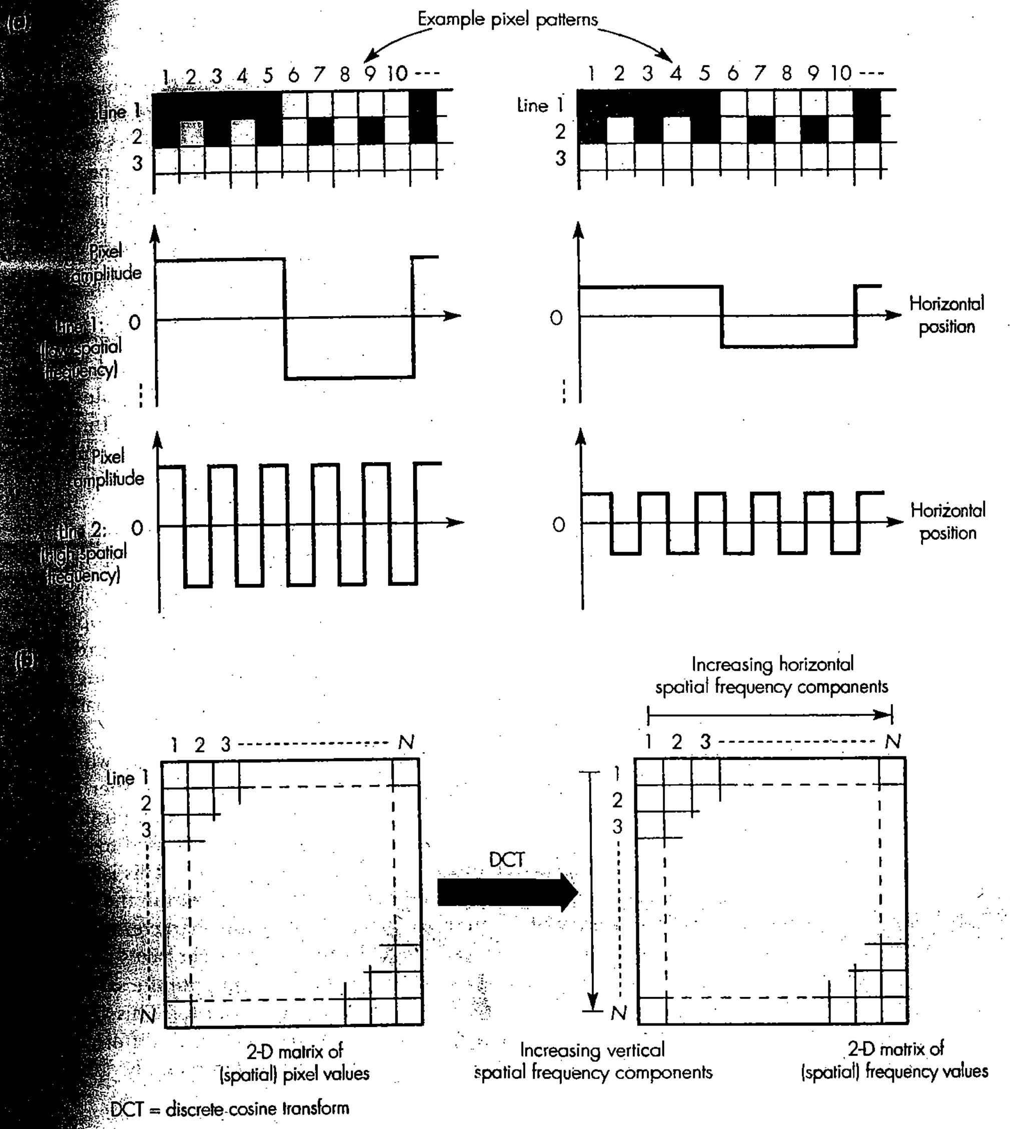

(a)
Example pixel patterns
1 2 3 4 5 6 7 8 9 10 ---
Line 1
2
3
Pixel amplitude
0
Line 1: (low spatial frequency)
Horizontal position
Line 2: (high spatial frequency)
(b)
Increasing horizontal spatial frequency components
1 2 3 ---- N
Line 1
N
DCT
2-D matrix of (spatial) pixel values
Increasing vertical spatial frequency components
2-D matrix of (spatial) frequency values
DCT = discrete cosine transform

Figure 3.2 Transform coding: (a) example pixel patterns; (b) DCT transform principles.

3.3 Text compression

As we saw in Section 2.3, the three different types of text – **unformatted, f** matted, and hypertext – are all represented as strings of **characters selec** from a defined set. The strings comprise alphanumeric **characters which a** interspersed with additional control characters. The different **types of te** use and interpret the latter in different ways. As we can deduce **from this, a** compression algorithm associated with text must be lossless since the loss o just a single character could modify the meaning of a complete string. In ge eral, therefore, we are restricted to the use of entropy encoding and, i practice, statistical encoding methods.

Essentially, there are two types of statistical encoding methods which are used with text: one which uses single characters as the basis of deriving an optimum set of codewords and the other which uses **variable-length** strings of characters. Two examples of the former are the Huffman and arithmetic coding algorithms and an example of the latter is the **Lempel-Ziv (LZ) algorithm**. We shall describe the principles of each of these algorithms in this section.

There are two types of coding used for text. The first is intended for applications in which the text to be compressed has known characteristics in terms of the characters used and their relative frequencies of occurrence. Using this information, instead of using fixed-length codewords, an optimum set of variable-length codewords is derived with the shortest codewords used to represent the most frequently occurring characters. The resulting set of codewords are then used for all subsequent transfers involving this particular type of text. This approach is known as **static coding**.

The second type is intended for more general applications in which the type of text being transferred may vary from one transfer to another. In this case the optimum set of codewords is also likely to vary from one transfer to another. To allow for this possibility, the codeword set that is used to transfer a particular text string is derived as the transfer takes place. This is done by building up knowledge of both the characters that are present in the text and their relative frequency of occurrence dynamically as the characters are being transmitted. Hence the codewords used change as a transfer takes place, but in such a way that the receiver is able to dynamically compute the same set of codewords that are being used at each point during a transfer. This approach is known as **dynamic** or **adaptive coding** and, since each uses a different algorithm to derive the codeword set, we shall describe each separately.

3.3.1 Static Huffman coding

With static Huffman coding the character string to be transmitted is first analyzed and the character types and their relative frequency determined. The coding operation involves creating an unbalanced tree with some branches (and hence codewords, in practice) shorter than others. The degree of imbalance is a function of the relative frequency of occurrence of the characters: the larger the spread, the more unbalanced is the tree. The resulting tree is known as the **Huffman code tree**.

A Huffman (code) tree is a **binary tree** with branches assigned the value 0 or 1. The base of the tree, normally the geometric top in practice, is known as the **root node** and the point at which a branch divides, a **branch node**. The termination point of a branch is known as a **leaf node** to which the symbols being encoded are assigned. An example of a Huffman code tree is shown in Figure 3.3(a). This corresponds to the string of characters AAAABBCD.

As each branch divides, a binary value of 0 or 1 is assigned to each new branch: a binary 0 for the left branch and a binary 1 for the right branch. The codewords used for each character (shown in the leaf nodes) are determined by tracing the path *from* the root node out to each leaf and forming a

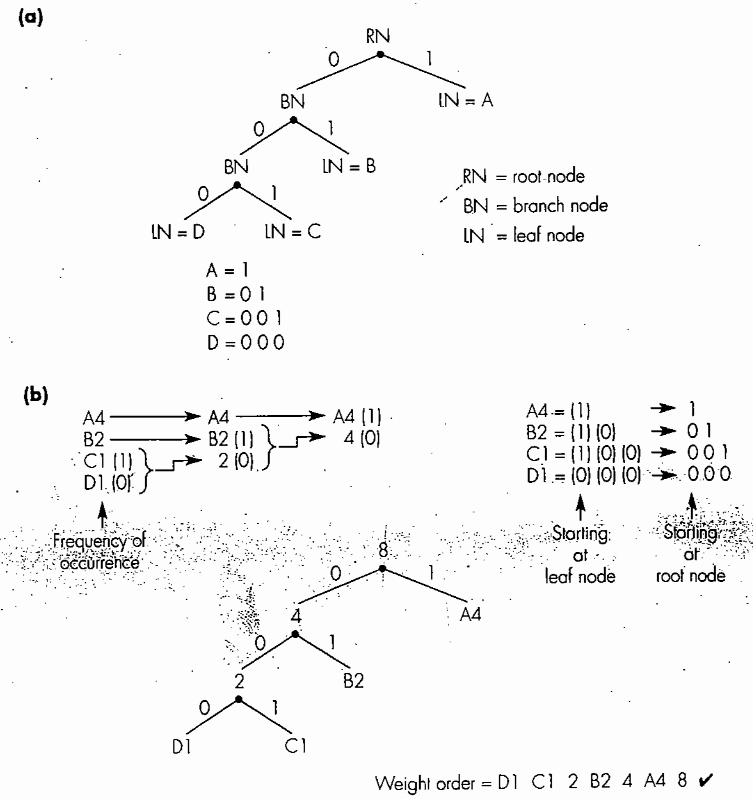


Figure 3.3 Huffman code tree construction: (a) final tree with codes; (b) tree derivation.

string of the binary values associated with each branch traced. We can ded
from the set of codes associated with this tree that it would take

$$4 \times 1 + 2 \times 2 + 1 \times 3 + 1 \times 3 = 14 \text{ bits}$$

to transmit the complete string AAAABBCD.

To illustrate how the Huffman code tree in Figure 3.3(a) is determin we must add information concerning the frequency of occurrence of ea character. Figure 3.3(b) shows the characters listed in a column in decreasi (weight) order. We derive the tree as follows.

The first two leaf nodes at the base of the list – C1 and D1 – are assign to the (1) and (0) branches respectively of a branch node. The two leaf nod are then replaced by a branch node whose weight is the sum of the weights o the two leaf nodes; that is, two. A new column is then formed containing th new branch node combined with the remaining nodes from the first colum again arranged in their correct weight order. This procedure is repeated un only two nodes remain.

To derive the resulting codewords for each character, we start with th character in the first column and then proceed to list the branch numbers – or 1 – as they are encountered. Thus for character A the first (and onl branch number is (1) in the last column while for C the first is (1) then (0) branch node 2 and finally (0) at branch node 4. The actual codewords, ho ever, start at the root and not the leaf node hence they are the reverse o these bit sequences. The Huffman tree can then be readily constructed fro the set of codewords.

We check that this is the optimum tree – and hence set of codewords – b listing the resulting weights of all the leaf and branch nodes in the tree sta ing with the smallest weight and proceeding from left to right and fro bottom to top. The codewords are optimum if the resulting list increments i weight order.

Example 3.2

[illegible]

Continued

[illegible] number of bits per character [illegible] with:

[illegible] the message (Shannon's value) [illegible]

[illegible] binary codewords

[illegible] ASCII codewords.

[illegible] formula states

[illegible] $= -\sum P_i \log_2 P_i$ bits per codeword

[illegible] $(0.25 \log_2 0.25) + 2(0.14 \log_2 0.14) + 4(0.055 \log_2 0.055)$ [illegible]

[illegible] = 2.175 bits per codeword

[illegible] of the codeword set using Huffman coding [illegible] 3.4(a). The characters are first listed in weight order and [illegible] characters at the bottom of the list are assigned to the [illegible] branches. Note that in this case, however, when the two [illegible] combined, the weight of the resulting branch node [illegible] is greater than the weight of the two characters E and F. [illegible] Hence the branch node is inserted into the second [illegible] both of these. The same procedure then repeats [illegible] only two entries in the list remaining.

[illegible] code tree corresponding to the derived set of codewords [illegible] shown in Figure 3.4(b) and, as we can see, this is the [illegible] all leaf and branch nodes [illegible]

[illegible] number of bits per codeword using Huffman coding [illegible]

[illegible] $2(2 \times 0.14) + 4(4 \times 0.055)$ [illegible]

[illegible] of the Shannon value

[illegible] length binary codewords:

[illegible] 8 characters – A through H – and hence [illegible] codeword is sufficient which is 90.7% of the Huffman value.

Using 7-bit ASCII codewords:

7 bits per codeword
which is 38.86% of the Huffman value.

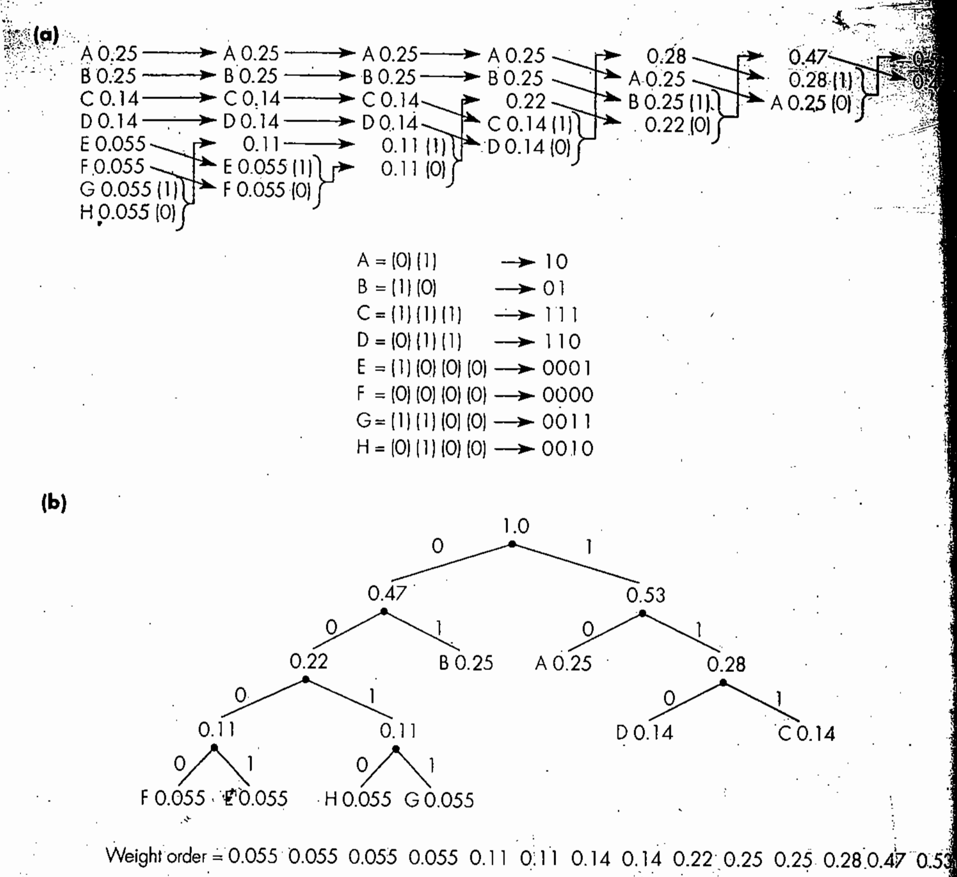

(a)
A 0.25
B 0.25
C 0.14
D 0.14
E 0.055
F 0.055
G 0.055 (1)
H 0.055 (0)
E 0.055 (1)
F 0.055 (0)
0.11
0.11 (1)
0.11 (0)
0.22
C 0.14 (1)
D 0.14 (0)
0.28
A 0.25
B 0.25 (1)
0.22 (0)
0.47
0.28 (1)
A 0.25 (0)
A = (0) (1) → 10
B = (1) (0) → 01
C = (1) (1) (1) → 111
D = (0) (1) (1) → 110
E = (1) (0) (0) (0) → 0001
F = (0) (0) (0) (0) → 0000
G = (1) (1) (0) (0) → 0011
H = (0) (1) (0) (0) → 0010
(b)
1.0
0
1
0.47
0.53
0.22
B 0.25
A 0.25
0.28
0.11
0.11
D 0.14
C 0.14
F 0.055
E 0.055
H 0.055
G 0.055
Weight order = 0.055 0.055 0.055 0.055 0.11 0.11 0.14 0.14 0.22 0.25 0.25 0.28 0.47 0.53

Figure 3.4 Huffman encoding example: (a) codeword generation; (b) Huffman code tree.

Since each character in its encoded form has a variable number of the received bitstream must be interpreted (decoded) in a bit-oriented rather than on fixed 7/8 bit boundaries. Because of the order in which are assigned during the encoding procedure, however, Huffman codewo have the unique property that a shorter codeword will never form start of a longer codeword. If, say, 011 is a valid codeword, then th cannot be any longer codewords starting with this sequence. We can co firm this by considering the codes derived in the earlier examples Figures 3.3 and 3.4.

This property, known as the **prefix property**, means that the receive bitstream can be decoded simply by carrying out a recursive search bit b

---

bit until each valid codeword is found. A flowchart of a suitable decoding algorithm is given in Figure 3.5(a). The algorithm assumes a table of codewords is available at the receiver and this also holds the corresponding ASCII codeword. The received bit stream is held in the variable BITSTREAM and the variable CODEWORD is used to hold the bits in each codeword while it is being constructed. As we can deduce from the flowchart, once a codeword is identified the corresponding ASCII codeword is written into the variable RECEIVE_BUFFER. The procedure repeats until all the bits in the received string have been processed. An example of a decoded string corresponding to the codeword set derived in Figure 3.3 is given in Figure 3.5(b).

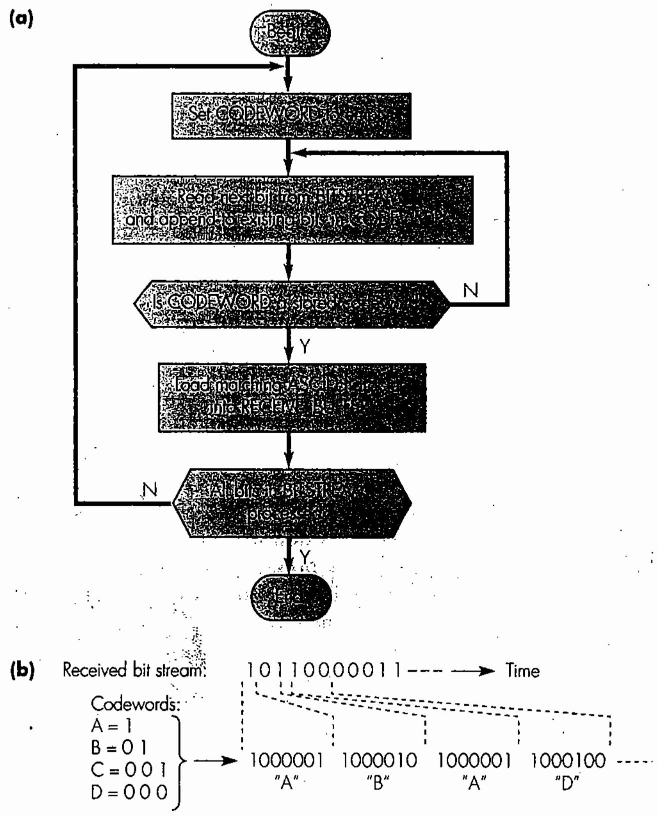

(a)
Begin
Set CODEWORD
Read next bit from
and append to existing bits in
Is CODEWORD
N
Y
Load matching ASCII
into RECEIVE
All bits
processed
N
Y
(b)
Received bit stream: 10110000011 --- → Time
Codewords:
A = 1
B = 01
C = 001
D = 000
1000001 1000010 1000001 1000100 ----
"A" "B" "A" "D"

Figure 3.5 Decoding of a received bitstream assuming codewords derived in Figure 3.3: (a) decoding algorithm; (b) example.

As the Huffman code tree (and hence codewords) varies for differe sets of characters being transmitted, for the receiver to perform the decodi operation it must know the codewords relating to the data being transmitte This can be done in two ways. Either the codewords relating to the next se data are sent before the data is transmitted, or the receiver knows in advan what codewords are being used.

The first approach leads to a form of adaptive compression since th codewords can be changed to suit the type of data being transmitted. The d advantage is the overhead of having to send the new set of codewords (an corresponding characters) whenever a new type of data is to be sent. An alte native is for the receiver to have one or more different sets of codewords an for the sender to indicate to the receiver (through an agreed message) whic codeword set to use for the next set of data.

For example, since a common requirement is to send text files generate by a word processor (and hence containing normal textual information detailed statistical analyses have been carried out into the frequency of occu rence of the characters in the English alphabet in normal written text. Th information has been used to construct the Huffman code tree for the alpha bet. If this type of data is being sent, the transmitter and receive automatically use this set of codewords. Other common data sets have bee analyzed in a similar way and, for further examples, you may wish to consu the bibliography at the end of the book.

3.3.2 Dynamic Huffman coding

The basic Huffman coding method requires both the transmitter and th receiver to know the table of codewords relating to the data being transmit ted. With dynamic Huffman coding, however, the transmitter (encoder) and receiver (decoder) build the Huffman tree – and hence codeword table dynamically as the characters are being transmitted/received.

With this method, if the character to be transmitted is currently presen in the tree its codeword is determined and sent in the normal way. If th character is not present – that is, it is its first occurrence – the character i transmitted in its uncompressed form. The encoder updates its Huffman tre either by incrementing the frequency of occurrence of the transmitted cha acter or by introducing the new character into the tree.

Each transmitted codeword is encoded in such a way that the receive in addition to being able to determine the character that is received, ca also carry out the same modifications to its own copy of the tree so that i can interpret the next codeword received according to the new updated tree structure.

To describe the details of the method, assume that the data (file) to be transmitted starts with the following character string:

This is simple ...

The steps taken by the transmitter are shown in Figure 3.6(a–g).

Both transmitter and receiver start with a tree that comprises the root node and a single **empty leaf node** – a leaf node with a zero frequency of occurrence – assigned to its 0-branch. There is always just one empty leaf node in the tree and its position – and codeword – varies as the tree is being constructed. It is represented in Figure 3.6 as e0.

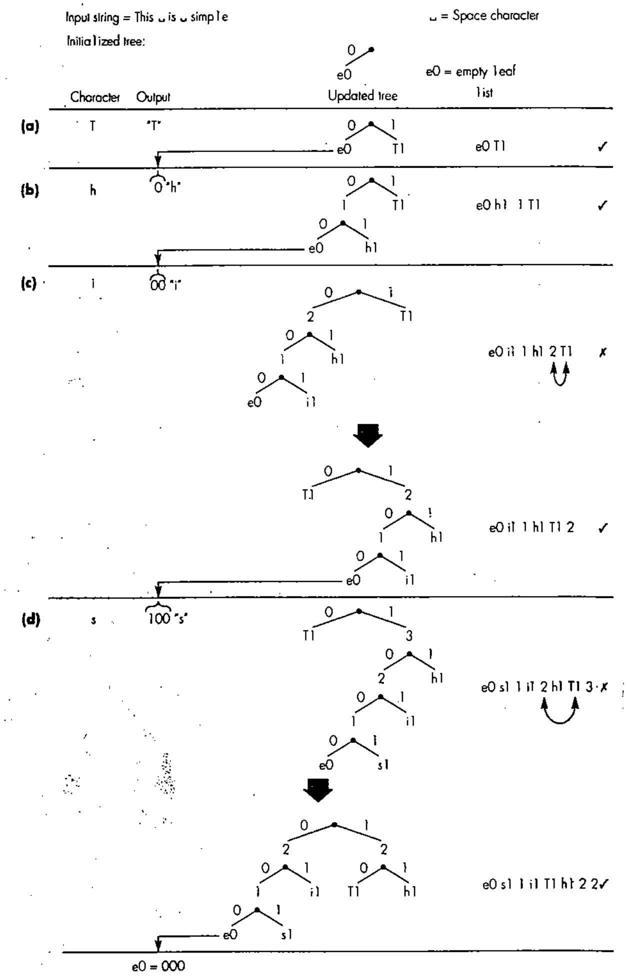


Figure 3.6 Dynamic Huffman encoding algorithm.

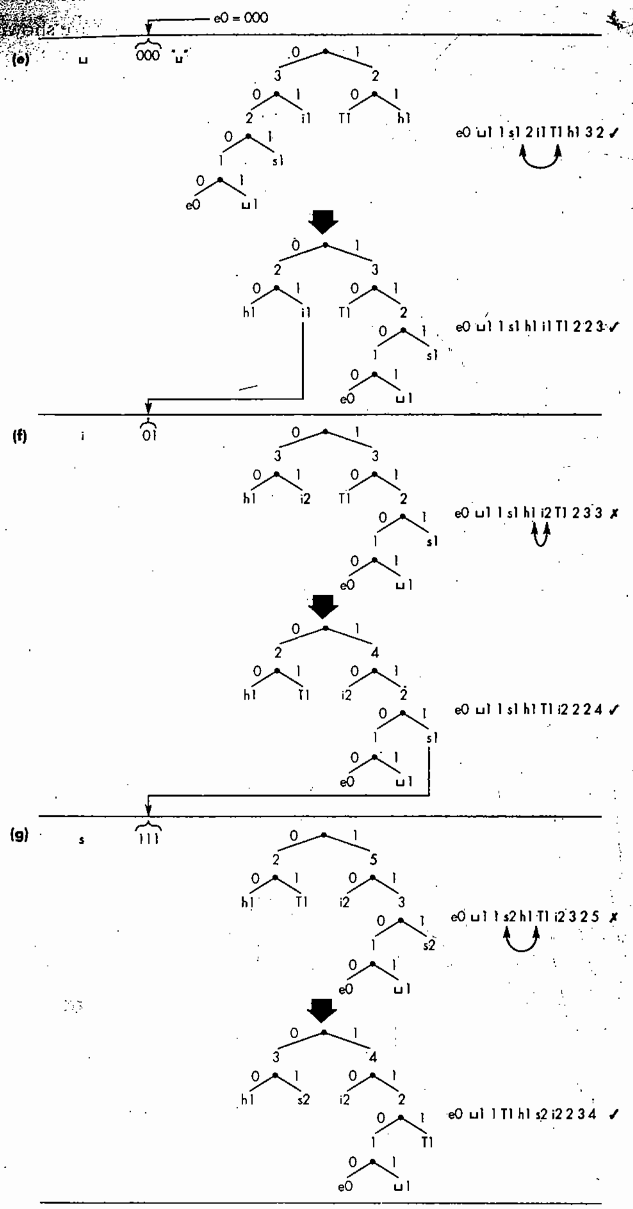

e0 = 000
(e)
000
(f)
01
(g)
111
e0 ⊔1 1 s1 2 i1 T1 h1 3 2 ✓
e0 ⊔1 1 s1 h1 i1 T1 2 2 3 ✓
e0 ⊔1 1 s1 h1 i2 T1 2 3 3 ✗
e0 ⊔1 1 s1 h1 T1 i2 2 2 4 ✓
e0 ⊔1 1 s2 h1 T1 i2 3 2 5 ✗
e0 ⊔1 1 T1 h1 s2 i2 2 3 4 ✓

Figure 3.6 Continued

The encoder then starts by reading the first character **T** and, since the tree is empty, it sends this in its uncompressed – say, ASCII – form. This is shown as "T" in the figure. The character is then assigned to the 1-branch of the root and, since this is the first occurrence of this character, it is shown as T1 in the tree. On reception, since the decoder's tree is also empty, it interprets the received bit string as an uncompressed character and proceeds to assign the character to its tree in the same way (Figure 3.6(a)).

For each subsequent character, the encoder first checks whether the character is already present in the tree. If it is, then the encoder sends the current codeword for the character in the normal way, the codeword being determined by the position of the character in the tree. If it is not present, then the encoder sends the current codeword for the empty leaf – again determined by its position in the tree – followed by the uncompressed codeword for the character. Since the decoder has the same tree as the encoder, it can readily deduce from the received bit string whether it is the current codeword of a (compressed) character or that of the empty leaf followed by the character in its uncompressed form.

The encoder and decoder proceed to update their copy of the tree based on the last character that has been transmitted/received. If it is a new character, the existing empty leaf node in the tree is replaced with a new branch node, the empty leaf being assigned to the 0-branch and the character to the 1-branch (Figure 3.6(b)).

If the character is already present in the tree, then the frequency of occurrence of the leaf node is incremented by unity. On doing this, the position of the leaf node may not now be in the optimum position in the tree. Hence each time the tree is updated – either by adding a new character or by incrementing the frequency of occurrence of an existing character – both the encoder and decoder check, and if necessary modify, the current position of all the characters in the tree.

To ensure that both the encoder and decoder do this in a consistent way, they first list the weights of the leaf and branch nodes in the updated tree from left to right and from bottom to top starting at the empty leaf. If they are all in weight order, all is well and the tree is left unchanged. If there is a node out of order, the structure of the tree is modified by exchanging the position of this node with the other node in the tree – together with its branch and leaf nodes – to produce an incremented weight order. The first occurrence is in Figure 3.6(c) and other examples are in parts (d)–(g).

The steps followed when a character to be transmitted has previously been sent are shown in Figure 3.6(f). At this point, the character to be transmitted is **i** and when the encoder searches the tree, it determines that **i** is already present and transmits its existing codeword – 01. The encoder then increments the character's weight – frequency of occurrence – by unity to i2 and updates the position of the modified node as before. Another example is shown in Figure 3.6(g) when the character **s** is to be transmitted.

We can deduce from this example that the savings in transmission bandwidth start only when characters begin to repeat themselves. In practice, the savings with text files can be significant, and dynamic Huffman coding is now used in a number of communication applications that involve the transmission of text.

3.3.3 Arithmetic coding

As we can deduce from Examples 3.1 and 3.2, Huffman coding achieves the Shannon value only if the character/symbol probabilities are all integer powers of ½. Clearly, in many instances, this is not the case and hence the set of codewords produced are rarely optimum. In contrast, the codewords produced using arithmetic coding always achieve the Shannon value. Arithmetic coding, however, is more complicated than Huffman coding and so we shall limit our discussion of it to the basic static coding mode of operation.

To illustrate how the coding operation takes place, consider the transmission of a message comprising a string of characters with probabilities of:

$$\mathbf{e} = 0.3, \quad \mathbf{n} = 0.3, \quad \mathbf{t} = 0.2, \quad \mathbf{w} = 0.1, \quad . = 0.1$$

At the end of each character string making up a message, a known character is sent which, in this example, is a period . . When this is decoded at the receiving side, the decoder interprets this as the end of the string/message.

Unlike Huffman coding which was a separate codeword for each character, arithmetic coding yields a single codeword for each encoded string of characters. The first step is to divide the numeric range from 0 to 1 into a number of different characters present in the message to be sent – including the termination character – and the size of each segment by the probability of the related character. Hence the assignments for our set of five characters may be as shown in Figure 3.7(a).

As we can see, since there are only five different characters, there are five segments, the width of each segment being determined by the probability of the related character. For example, the character **e** has a probability of 0.3 and hence is assigned the range from 0.0 to 0.3, the character **n** – which also has a probability of 0.3 – the range from 0.3 to 0.6, and so on. Note, however, that an assignment in the range, say, 0.8 to 0.9, means that the probability in the cumulative range is from 0.8 to 0.8999... . Once this has been done, we are ready to start the encoding process. An example is shown in Figure 3.7(b) and, in this example, we assume the character string/message to be encoded is the single word **went.** .

The first character to be encoded **w** is in the range 0.8 to 0.9. Hence, as we shall see, the final (numeric) codeword is a number in the range 0.8 to 0.8999 ... since each subsequent character in the string subdivides the range 0.8 to 0.9 into progressively smaller segments each determined by the probabilities of the characters in the string.

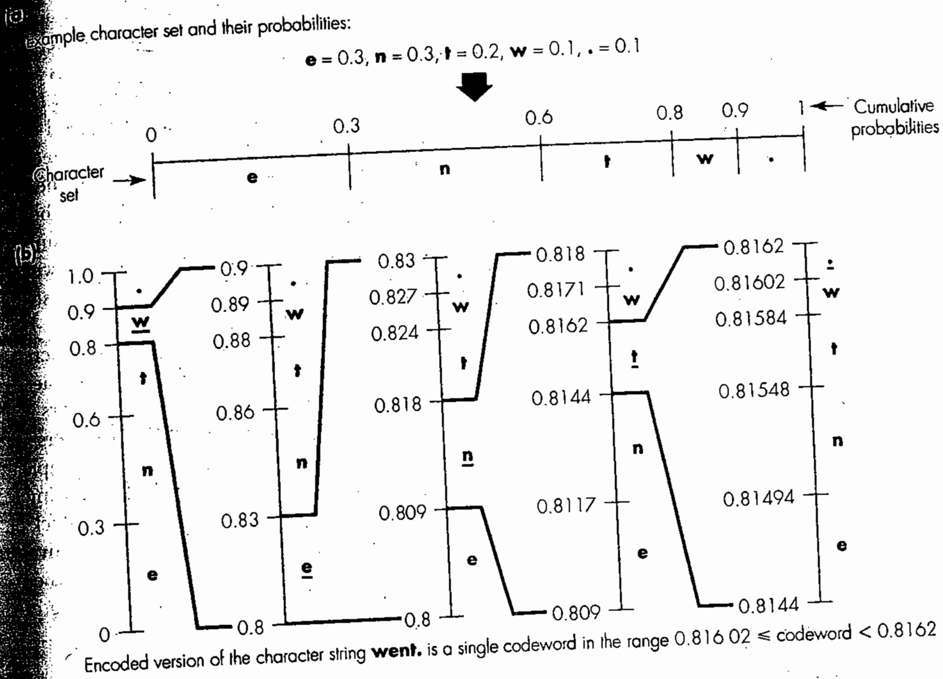


Figure 3.7 Arithmetic coding principles: (a) example character set and their range assignments; (b) encoding of the string went..

As we can see in the example, since **w** is the first character in the string, the range 0.8 to 0.9 is itself subdivided into five further segments, the width of each segment again determined by the probabilities of the five characters. Hence the segment for the character **e**, for example, is from 0.8 to 0.83 (0.8 + 0.3 × 0.1), the character **n** from 0.83 to 0.86 (0.83 + 0.3 × 0.1), and so on.

The next character in the string is **e** and hence its range (0.8 to 0.83) is again subdivided into five segments. With the new assignments, therefore, the character **e** has a range from 0.8 to 0.809 (0.8 + 0.3 × 0.03), the character **n** from 0.809 to 0.818 (0.809 + 0.3 × 0.03), and so on. This procedure continues until the termination character **.** is encoded. At this point, the segment range of **.** is from 0.81602 to 0.8162 and hence the codeword for the complete string is any number within the range:

$$0.81602 \leq \text{codeword} > 0.8162$$

In the static mode, the decoder knows the set of characters that ar sent in the encoded messages it receives as well as the segment to which character has been assigned and its related range. Hence with this as a point, the decoder can follow the same procedure as that followed b encoder to determine the character string relating to each received c word. For example, if the received codeword is, say, 0.8161, then the dec can readily determine from this that the first character is **w** since it is the character within the range 0.8 to 0.9. It then expands this interval as be and determines that the second character must be **e** since 0.8161 is within range 0.8 to 0.83. This procedure then repeats until it decodes the known mination character . at which point it has recreated the, say, ASCII st relating to **went.** and passes this on for processing.

As we can deduce from this simple example, the number of deci digits in the final codeword increases linearly with the number of charac in the string to be encoded. Hence the maximum number of characters i string is determined by the precision with which floating-point numbers represented in the source and destination computers. As a result, a comp message may be first fragmented into multiple smaller strings. Each string then encoded separately and the resulting set of codewords sent as a block (binary) floating-point numbers each in a known format. Alternative **binary arithmetic coding** can be used but, as we indicated earlier, this is o side the scope of the book. Further details relating to arithmetic coding c be found in the bibliography for this chapter at the end of the book.

3.3.4 Lempel–Ziv coding

The Lempel–Ziv (LZ) compression algorithm, instead of using single char ters as the basis of the coding operation, uses strings of characters. F example, for the compression of text, a table containing all the possible ch acter strings – for example words – that occur in the text to be transferred held by both the encoder and decoder. As each word occurs in the te instead of sending the word as a set of individual – say, ASCII – codewo the encoder sends only the index of where the word is stored in the ta and, on receipt of each index, the decoder uses this to access the correspo ing word/string of characters from the table and proceeds to reconstruct text into its original form. Thus the table is used as a dictionary and the algorithm is known as a **dictionary-based** compression algorithm.

Most word-processing packages have a dictionary associated with the which is used for both spell checking and for the compression of te Typically, they contain in the region of 25 000 words and hence 15 bits which has 32 768 combinations – are required to encode the index. To sen the word "multimedia" with such a dictionary would require just 15 bit instead of 70 bits with 7-bit ASCII codewords. This results in a compression ratio of 4.7:1. Clearly, shorter words will have a lower compression ratio and longer words a higher ratio.

ample 3.3

The LZ algorithm is to be used to compress a text file prior to its trans mission. If the average number of characters per word is 6 and the dictionary used contains 4096 words, derive the average compression ratio that is achieved relative to using 7-bit ASCII codewords.

Answer:

In general, a dictionary with an index of n bits can contain up to 2^n entries. Now $4096 = 2^{12}$ and hence an index of 12 bits is required.

Using 7-bit ASCII codewords and an average of 6 characters per word requires 42 bits.

Hence compression ratio = 42/12 = 3.5:1.

As with the other static coding methods, the basic requirement with the LZ algorithm is that a copy of the dictionary is held by both the encoder and the decoder. Although this is acceptable for the transmission of text which has been created using a standard word-processing package, it can be relatively inefficient if the text to be transmitted comprises only a small subset of the words stored in the dictionary. Hence a variation of the LZ algorithm has been developed which allows the dictionary to be built up dynamically by the encoder and decoder as the compressed text is being transferred. In this way, the size of the dictionary is often a better match to the number of different words in the text being transmitted than if a standard dictionary was used.

3.3.5 Lempel–Ziv–Welsh coding

The principle of the Lempel–Ziv–Welsh (LZW) coding algorithm is for the encoder and decoder to build the contents of the dictionary dynamically as the text is being transferred. Initially, the dictionary held by both the encoder and decoder contains only the character set – for example ASCII – that has been used to create the text. The remaining entries in the dictionary are then built up dynamically by both the encoder and decoder and contain the words that occur in the text. For example, if the character set comprises 128 characters and the dictionary is limited to, say, 4096 entries, then the first 128 entries would contain the single characters that make up the character set and the remaining 3968 entries would each contain strings of two or more characters that make up the words in the text being transferred. As we can see, the more frequently the words stored in the dictionary occur in the text, the higher the level of compression.

In order to describe how the dictionary is built up, let us assume that the text to be compressed starts with the string:

This is simple as it is ...

Since the idea is for the dictionary to contain only words, only strings of characters that consist of alphanumeric characters are st in the dictionary and all the other characters in the set are interpret word delimiters.

Initially, the dictionary held by both the encoder and decoder con only the individual characters from the character set being used; for exam the 128 characters in the ASCII character set. Hence the first word in example text is sent by the encoder using the index of each of the four c acters *T*, *h*, *i*, and *s*. At this point, when the encoder reads the next chara from the string – the first space (SP) character – it determines that this is an alphanumeric character. It therefore transmits the character using index as before but, in addition, interprets it as terminating the first w and hence stores the preceding four characters in the next available (f location in the dictionary. Similarly the decoder, on receipt of the first indices/codewords, reads the character stored at each index and comme to reconstruct the text. When it determines that the fifth character is a sp character, it interprets this as a word delimiter and proceeds to store the wo *This* in its dictionary.

The same procedure is followed by both the encoder and decoder transferring the other words except the encoder, prior to sending each wo in the form of single characters, first checks to determine if the word is c rently stored in its dictionary and, if it is, it sends only the index for the wo Similarly the decoder, since it also has the word stored in its dictionary, u the index to access the string of characters that make up the word. So w the example text string, after the space character following the second occ rence of the word *is*, the contents of the dictionary held by both the enco and the decoder will be as shown in Figure 3.8(a). As we can see, since thi the second occurrence of the word *is*, it is transferred using only the index where it is stored in the dictionary (129).

As we can deduce from this example, a key issue in determin the level of compression that is achieved, is the number of entries in t dictionary since this, in turn, determines the number of bits that a required for the index. With a static dictionary, the number of entrie fixed and, for the example we identified earlier, a dictionary contain 25 000 words requires 15 bits to encode the index. When building dictionary dynamically, however, the question arises as to how many entri should be provided for the dictionary. Clearly, if too few entries are provide then the dictionary will contain only a subset of the words that occur the text while if too many are provided, then it will contain empty space which, in turn, makes the index unnecessarily long. In order to optimiz the number of bits used for the index, at the commencement of eac transfer the number of entries is set to a relatively low value but, should th available space become full, then the number of entries is allowed t increase incrementally.

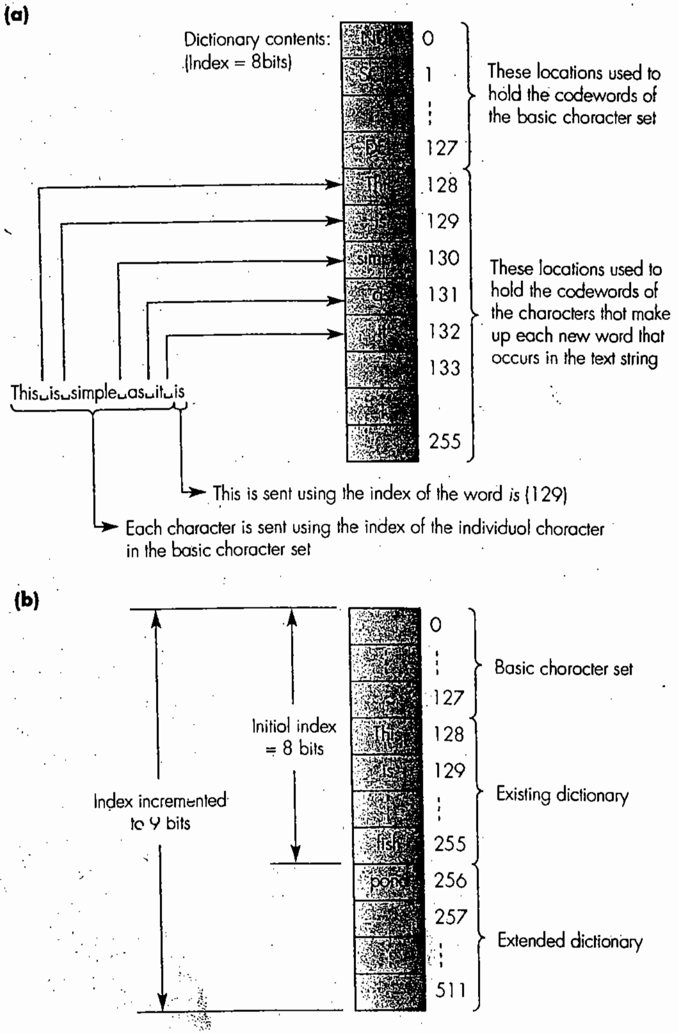

(a)
Dictionary contents:
(Index = 8bits)
0
1
127
128
129
130
131
132
133
255
These locations used to hold the codewords of the basic character set
These locations used to hold the codewords of the characters that make up each new word that occurs in the text string
This␣is␣simple␣as␣it␣is
This is sent using the index of the word *is* (129)
Each character is sent using the index of the individual character in the basic character set
(b)
Initial index = 8 bits
Index incremented to 9 bits
0
127
128
129
255
256
257
511
This
is
fish
pond
Basic character set
Existing dictionary
Extended dictionary

Figure 3.8 LZW compression algorithm: (a) basic operation; (b) dynamically extending the number of entries in the dictionary.

For example, in an application that uses 128 characters in the basic character set, then both the encoder and decoder would start with, say, 256 entries in the dictionary. This requires an index/codeword length of 8 bits and the dictionary would provide space for the 128 characters in the character set and a further 128 locations for words that occur in th Should this number of locations become insufficient, on detecting thi the encoder and decoder would double the size of their dictionary t locations. Clearly, this necessitates an index length of 9 bits and so fro point, the encoder uses 9-bit codewords. However, since the decoder ha doubled the size of its own directory, it expects 9-bit codewords fro point. In this way, the number of entries in the dictionary more accu reflects the number of different words in the text being transferre hence optimizes the number of bits used for each index/codeword.

The procedure is shown in diagrammatic form in Figure 3.8(b). I example it is assumed that the last entry in the existing table at location 2 the word *fish* and the next word in the text that is not currently in the di nary is *pond*.

3.4 Image compression

Recall from Section 2.4 how images can be of two basic types: compu generated (also known as graphical) images and digitized images (of b documents and pictures). Although both types are displayed (and printed the form of a two-dimensional matrix of individual picture elements, n mally a graphical image is represented differently in the computer file syst Typically, this is in the form of a program (written in a particular graph programming language) and, since this type of representation requires c siderably less memory (and hence transmission bandwidth) than t corresponding matrix of picture elements, whenever possible, graphics a transferred across a network in this form. In the case of digitized docume and pictures, however, once digitized, the only form of representation is a two-dimensional matrix of picture elements.

In terms of compression, when transferring graphical images which a represented in their program form, a lossless compression algorithm mus used similar, for example, to those in the last section. However, when the g ated image/graphic is to be transferred across the network in its bit-m form, then this is normally compressed prior to its transfer. There a number of different compression algorithms and associated file forma use and we shall describe two of these in the next two sections.

To transfer digitized images a different type of compression algori must normally be employed and, in practice, two different schemes are us The first is based on a combination of run-length and statistical encodi Hence it is lossless and is used for the transfer of the digitized docume generated by scanners such as those used in facsimile machines. The seco is based on a combination of transform, differential, and run-length enco ing and has been developed for the compression of both bitonal and colo digitized pictures. Since there is an international standard associated wi both schemes, we shall limit our discussion to these two schemes.

3.4.1 Graphics interchange format

The **graphics interchange format** (GIF) is used extensively with the Internet for the representation and compression of graphical images. Although color images comprising 24-bit pixels are supported – 8 bits each for R, G, and B – GIF reduces the number of possible colors that are present by choosing the 256 colors from the original set of 2^{24} colors that match most closely those used in the original image. The resulting table of colors therefore consists of 256 entries, each of which contains a 24-bit color value. Hence instead of sending each pixel as a 24-bit value, only the 8-bit index to the table entry that contains the closest match color to the original is sent. This results in a compression ratio of 3:1. The table of colors can relate either to the whole image – in which case it is referred to as the **global color table** – or to a portion of the image, when it is referred to as a **local color table**. The contents of the table are sent across the network – together with the compressed image data and other information such as the screen size and aspect ratio – in a standardized format. The principles of the scheme are shown in Figure 3.9(a).

As we show in Figure 3.9(b), the LZW coding algorithm can be used to obtain further levels of compression. We described this earlier in Section 3.3.5 when we discussed text compression and, in the case of image compression, this works by extending the basic color table dynamically as the compressed image data is being encoded and decoded. As with text compression, the occurrence of common strings of pixel values – such as long strings of the same color – are detected and these are entered into the color table after the 256 selected colors. However in this application, since each entry in the color table comprises 24 bits, in order to save memory, to represent each string of pixel values just the corresponding string of 8-bit indices to the basic color table are used. If we limit each entry in the table to 24 bits, then this will allow common strings comprising three pixel values to be stored in each location of the extended table. Normally, since the basic table contains 256 entries, an initial table size of 512 entries is selected which allows for up to 256 common strings to be stored. As with text compression, however, should more strings be found, then the number of entries in the table is allowed to increase incrementally by extending the length of the index by 1 bit.

GIF also allows an image to be stored and subsequently transferred over the network in an interlaced mode. This can be useful when transferring images over either low bit rate channels or the Internet which provides a variable transmission rate. With this mode, the compressed image data is organized so that the decompressed image is built up in a progressive way as the data arrives. To achieve this, the compressed data is divided into four groups as shown in Figure 3.10 and, as we can see, the first contains 1/8 of the total compressed image data, the second a further 1/8, the third a further 1/4, and the last the remaining 1/2.

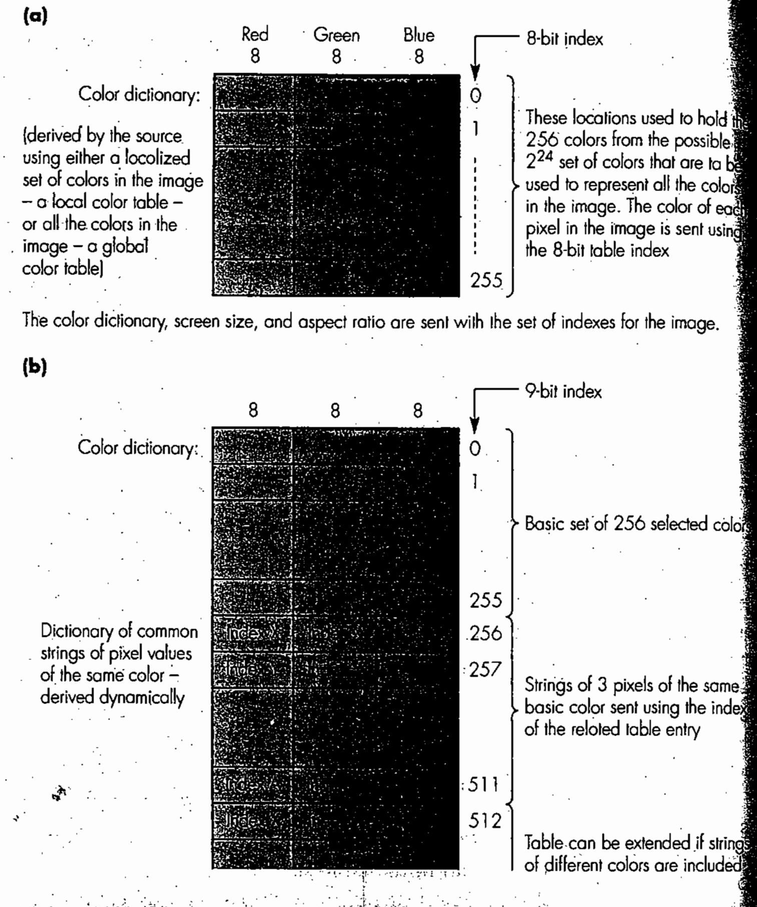

(a)
Red 8
Green 8
Blue 8
8-bit index
Color dictionary:
(derived by the source using either a localized set of colors in the image – a local color table – or all the colors in the image – a global color table)
0
1
255
These locations used to hold th 256 colors from the possible 2^24 set of colors that are to b used to represent all the colors in the image. The color of eac pixel in the image is sent using the 8-bit table index
The color dictionary, screen size, and aspect ratio are sent with the set of indexes for the image.
(b)
8
8
8
9-bit index
Color dictionary:
0
1
Basic set of 256 selected colo
255
256
257
Dictionary of common strings of pixel values of the same color – derived dynamically
Strings of 3 pixels of the same basic color sent using the index of the related table entry
511
512
Table can be extended if strings of different colors are included

Figure 3.9 GIF compression principles: (a) basic operational mode; (b) dynamic mode using LZW coding.

3.4.2 Tagged image file format

The **tagged image file format** (TIFF) is also used extensively. It supports pixe resolutions of up to 48 bits – 16 bits each for R, G, and B – and is intended fo the transfer of both images and digitized documents. The image data, therefore, can be stored – and hence transferred over the network – in a numbe of different formats. The particular format being used is indicated by a code number and these range from the uncompressed format (code number 1) through to LZW-compressed which is code number 5. Code numbers 2, 3, and 4 are intended for use with digitized documents. These use the same compression algorithms that are used in facsimile machines which we discuss in the next section.

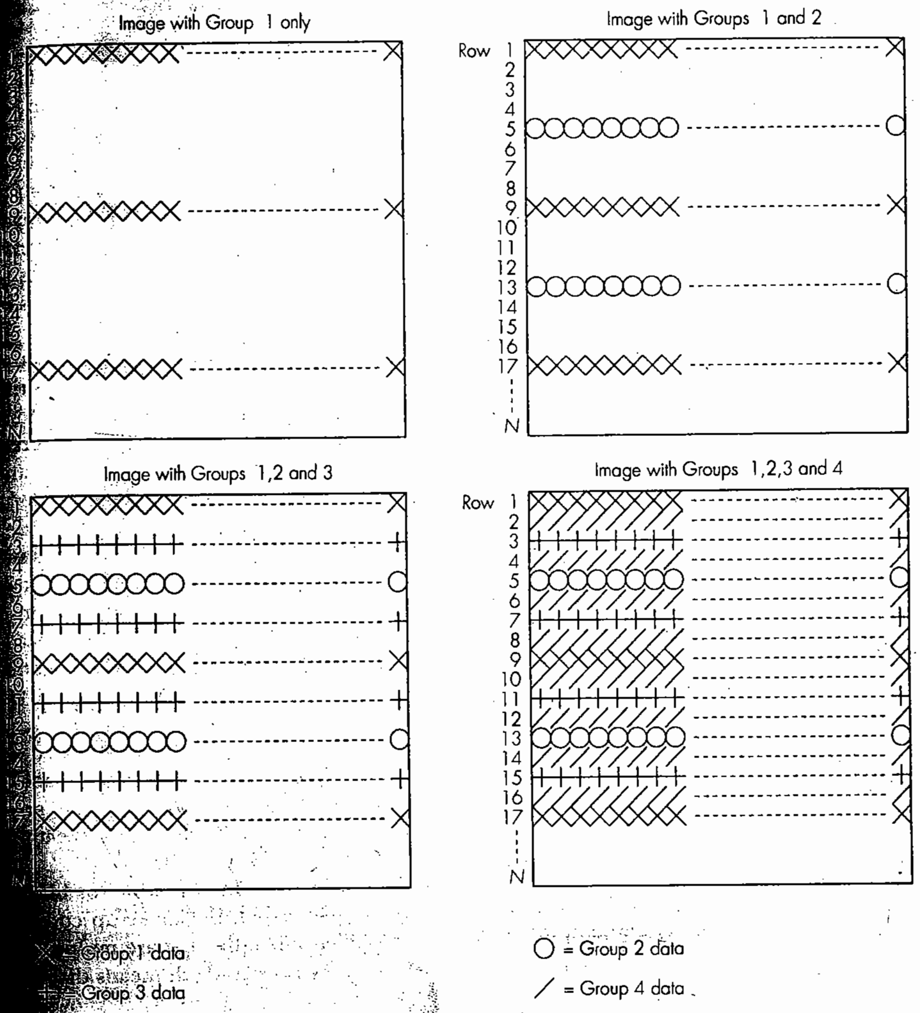


Image with Group 1 only
Image with Groups 1 and 2
Row
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
N
Image with Groups 1,2 and 3
Image with Groups 1,2,3 and 4
Group 1 data
Group 3 data
○ = Group 2 data
╱ = Group 4 data

3.10 GIF interlaced mode.

The LZW compression algorithm that is used is the same as that used with GIF. It starts with a basic color table containing 256 colors and the table can be extended to contain up to 4096 entries containing common strings of pixels in the image being transferred. Again, a standard format is used for the transfer of both the color table and the compressed image data.

3.4.3 Digitized documents

We described the principle of operation of the scanners used in facsim machines to digitize bitonal images (such as a printed document) in Secti 2.4.3. The digital representation of a scanned page was shown in Figu 2.11(b) and, even though only a single binary bit is used to represent ea picture element, with the resolutions used, this produces an uncompress bit stream of the order of 2 Mbits. In most cases this must be transferr using modems and the public switched telephone network. The relative low bit rates available with modems means that it would be both costly an time consuming to transfer a total document comprising many pages in th basic form.

With most documents, many scanned lines consist only of long strings o white picture elements – pels – while others comprise a mix of long strings o white and long strings of black pels. Since facsimile machines are normal used with public carrier networks, the ITU-T has produced standards relati to them. These are T2 (Group 1), T3 (Group 2), T4 (Group 3), and T (Group 4). The first two are earlier standards and are now rarely used. Th last two, however, both operate digitally; Group 3 with modems for use wit an analog PSTN, and Group 4 all-digital for use with digital networks such a the ISDN. Both use data compression, and compression ratios in excess o 10:1 are common with most document pages. The time taken to transmit page is reduced to less than a minute with Group 3 machines and, because o the added benefit of a higher transmission rate (64 kbps), to less than a fe seconds with a Group 4 machine.

As part of the standardization process, extensive analyses of typical scanne document pages were made. Tables of codewords were produced based on th relative frequency of occurrence of the number of contiguous white and bla pels found in a scanned line. The resulting codewords are fixed and groupe into two separate tables: the **termination-codes table** and the **make-up cod table**. The codewords in each table are shown in Figure 3.11.

Codewords in the termination-codes table are for white or black ru lengths of from 0 to 63 pels in steps of 1 pel; the make-up codes tab contains codewords for white or black run-lengths that are multiples of pels. A technique known as **overscanning** is used which means that all lin start with a minimum of one white pel. In this way, the receiver knows the fi codeword always relates to white pels and then alternates between black an white. Since the scheme uses two sets of codewords (termination and ma up) they are known as **modified Huffman codes**. As an example, a run-lengt of 12 white pels is coded directly as 001000. Similarly, a run-length of 12 blac pels is coded directly as 0000111. A run-length of 140 black pels, however, encoded as 000011001000 + 0000111; that is, 128 + 12 pels. Run-length exceeding 2560 pels are encoded using more than one make-up code plu one termination code.

There is no error-correction protocol with Group 3. From the list o codewords, we can deduce that if one or more bits is corrupted during its

(a)

White run-length	Codeword	Black run-length	Codeword
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	0010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101011	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

(b)

White run-length	Codeword	Black run-length	Codeword
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000001110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101
1792	00000001000	1792	00000001000
1856	00000001100	1856	00000001100
1920	00000001101	1920	00000001101
1984	000000010010	1984	000000010010
2048	000000010011	2048	000000010011
2112	000000010100	2112	000000010100
2176	000000010101	2176	000000010101
2240	000000010110	2240	000000010110
2304	000000010111	2304	000000010111
2368	000000011100	2368	000000011100
2432	000000011101	2432	000000011101
2496	000000011110	2496	000000011110
2560	000000011111	2560	000000011111
EOL	00000000001	EOL	00000000001

Figure 3.11 ITU–T Group 3 and 4 facsimile conversion codes: (a) termination-codes, (b) make-up codes.

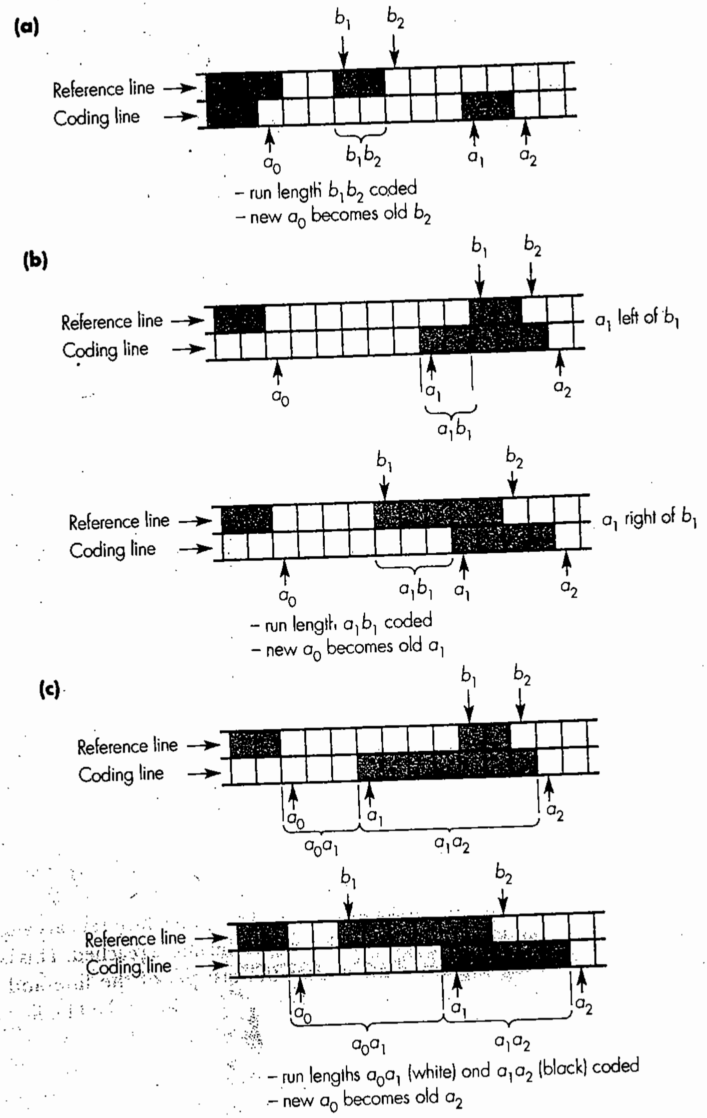
transmission through the network, the receiver will start to interpret sub quent codewords on the wrong bit boundaries. The receiver thus becom unsynchronized and cannot decode the received bit string. To enable th receiver to regain synchronism, each scanned line is terminated with a kno **end-of-line (EOL) code**. In this way, if the receiver fails to decode a valid co word after the maximum number of bits in a codeword have been scann (parsed), it starts to search for the EOL pattern. If it fails to decode an E after a preset number of lines, it aborts the reception process and informs th sending machine. A single EOL precedes the codewords for each scann page and a string of six consecutive EOLs indicates the end of each page.

Because each scanned line is encoded independently, the T4 codi scheme is known as a **one-dimensional coding** scheme. As we can conclude works satisfactorily providing the scanned image contains significant areas white or black pels which occur, for example, where documents consist of l ters and line drawings. Documents containing photographic image however, are not satisfactory as the different shades of black and white a represented by varying densities of black and white pels. This, in turn, resul in a large number of very short black or white run-lengths which, with the T coding scheme, can lead to a **negative compression ratio**; that is, more b are needed to send the scanned document in its compressed form than a needed in its uncompressed form.

For this reason the alternative T6 coding scheme has been defined. It is a optional feature in Group 3 facsimile machines but is compulsory in Group machines. When supported in Group 3 machines, the EOL code at the end each (compressed) line has an additional tag bit added. If this is a binary then the next line has been encoded using the T4 coding scheme, if it is then the T6 coding scheme has been used. The latter is known as **modifie** **modified READ (MMR) coding.** It is also known as **two-dimensional** or 2 **coding** since it identifies black and white run-lengths by comparing adjace scan lines. READ stands for **relative element address designate**, and it is "mo fied" since it is a modified version of an earlier (modified) coding scheme.

MMR coding exploits the fact that most scanned lines differ from the p vious line by only a few pels. For example, if a line contains a black-run th the next line will normally contain the same run plus or minus up to th pels. With MMR coding the run-lengths associated with a line are identi by comparing the line contents, known as the **coding line** (**CL**), relative to th immediately preceding line, known as the **reference line** (**RL**). We alw assume the first reference line to be an (imaginary) all-white line and the fi line proper is encoded relative to this. The encoded line then becomes th reference line for the following line, and so on. To ensure that the comple page is scanned, the scanner head always starts to the left of the page, so ea line always starts with an imaginary white pel.

We identify the run lengths associated with a coding line as one of thr possibilities or **modes** relative to the reference line. Examples of the thr modes are shown in Figure 3.12. The three modes are identified by the pos tion of the next run-length in the reference line (b_1b_2) relative to the sta


(a)
b_1
b_2
Reference line
Coding line
a_0
b_1b_2
a_1
a_2
– run length b_1b_2 coded
– new a_0 becomes old b_2
(b)
b_1
b_2
Reference line
Coding line
a_1 left of b_1
a_0
a_1
a_2
a_1b_1
b_1
b_2
Reference line
Coding line
a_1 right of b_1
a_0
a_1b_1
a_1
a_2
– run length a_1b_1 coded
– new a_0 becomes old a_1
(c)
b_1
b_2
Reference line
Coding line
a_0
a_1
a_2
a_0a_1
a_1a_2
b_1
b_2
Reference line
Coding line
a_0
a_1
a_2
a_0a_1
a_1a_2
– run lengths a_0a_1 (white) and a_1a_2 (black) coded
– new a_0 becomes old a_2

Note: a_0 is the first pel of a new codeword and can be black or white
a_1 is the first pel to the right of a_0 with a different color
b_1 is the first pel on the reference line to the right of a_0 with a different color
b_2 is the first pel on the reference line to the right of b_1 with a different color

Figure 3.12 **Some example run-length possibilities: (a) pass mode; (b) vertical mode; (c) horizontal mode.**

and end of the next pair of run-lengths in the coding line (a_0a_1 and Note that the same procedure is used to encode the runs of both bla white pels. The three possibilities are:

1. **Pass mode:** This is the case when the run-length in the reference li (b_1b_2) is to the left of the next run-length in the coding line (a_1a_2), t is, b_2 is to the left of a_1. An example is given in Figure 3.12(a) and, f this mode, the run-length b_1b_2 is coded using the codewords given i Figure 3.11. Note that if the next pel on the coding line, a_1, is direct below b_2 then this is not pass mode.
2. **Vertical mode:** This is the case when the run-length in the reference (b_1b_2) overlaps the next run-length in the coding line (a_1a_2) by a maximum of plus or minus 3 pels. Two examples are given in Figure 3.12(b) and, for this mode, just the difference run-length a_1b_1 is code Most codewords are in this category.
3. **Horizontal mode:** This is the case when the run-length in the referenc line (b_1b_2) overlaps the run-length (a_1a_2) by more than plus or minus 3 pels. Two examples are given in Figure 3.12(c) and, for this mode, the run-lengths a_0a_1 and a_1a_2 are coded using the codewords in Figure 3.1

A flowchart of the coding procedure is shown in Figure 3.13. Note the first a_0 is set to an imaginary white pel *before* the first pel of the line hence the first a_0a_1 run-length will be $a_0a_1 - 1$. If during the coding of a a_1, a_2, b_1, or b_2 are not detected, then they are set to an imaginary pel tioned immediately after the last pel on the respective line.

Once the first/next position of a_0 has been determined, the positions o a_2, b_1, and b_2 for the next codeword are located. The mode is then determi by computing the position of b_2 relative to a_1. If it is to the left, this indicates mode. If it is not to the left, then the magnitude of a_1b_1 is used to deter whether the mode is vertical or horizontal. The codeword for the iden mode is then computed and the start of the next codeword position, a_0, up to the appropriate position. This procedure repeats alternately between and black runs until the end of the line is reached. This is an imaginary pe tioned immediately after the last pel of the line and is assumed to different color from the last pel. The current coding line then becomes th reference line and the next scanned line the new coding line.

Since the coded run-lengths relate to one of the three modes, additio codewords are used either to indicate to which mode the following co word(s) relate – pass or horizontal – or to specify the length of the codew directly – vertical. The additional codewords are given in a third table kno as the **two-dimensional code table**. Its contents are as shown in Table 3.1. T final entry in the table, known as the **extension mode**, is a unique codew that aborts the encoding operation prematurely before the end of the pa This is provided to allow a portion of a page to be sent in its uncompress form or possibly with a different coding scheme.

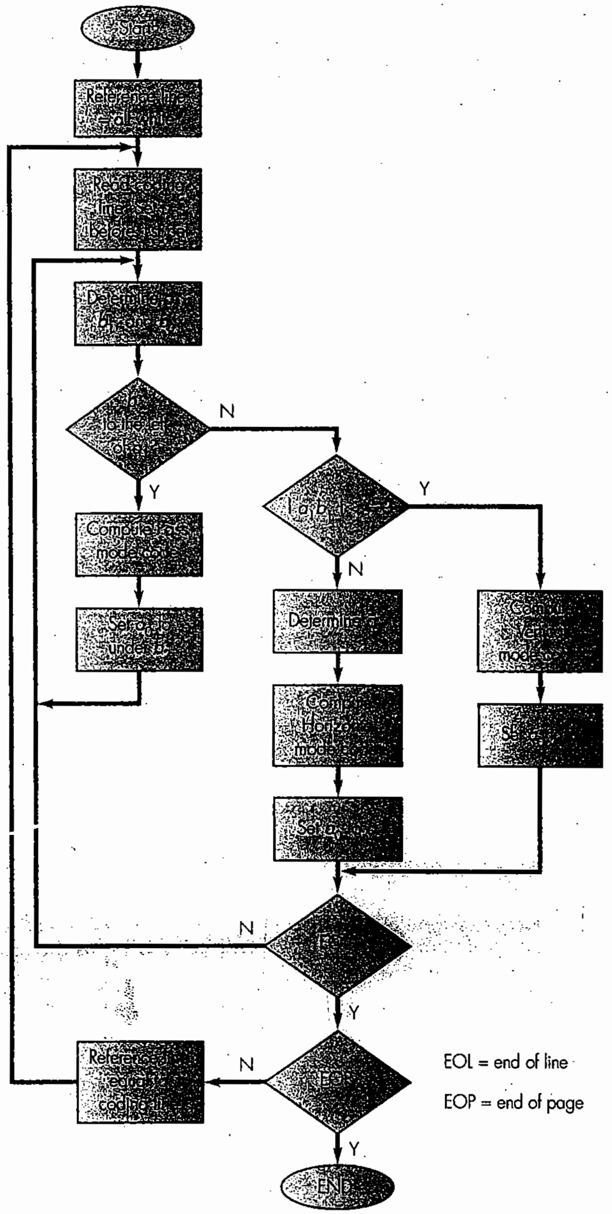


Figure 3.13 Modified-modified READ coding procedure.

Table 3.1 Two-dimensional code table contents.

Mode	Run-length to be encoded	Abbreviation	Codeword
Pass	b_1b_2	P	0001
Horizontal	a_0a_1, a_1a_2	H	001 + $M(a_0a_1)$ + $M(a_1a_2)$
Vertical	$a_1b_1 = 0$	$V(0)$	1
	$a_1b_1 = -1$	$V_R(1)$	011
	$a_1b_1 = -2$	$V_R(2)$	000011
	$a_1b_1 = -3$	$V_R(3)$	0000011
	$a_1b_1 = +1$	$V_L(1)$	010
	$a_1b_1 = +2$	$V_L(2)$	000010
	$a_1b_1 = +3$	$V_L(3)$	0000010
Extension			0000001000

3.4.4 Digitized pictures

We described the digitization of both continuous-tone monochromatic pictures and color pictures in Section 2.4.3. We also calculated the amount of computer memory required to store and display these pictures on a number of popular types of display and tabulated these in Table 2.1. The amount of memory ranged from (approximately) 307 kbytes through to 2.4 Mbytes and, as we concluded, all would result in unacceptably long delays in most interactive applications that involve low bit rate networks.

In order to reduce the time to transmit digitized pictures, compression is normally applied to the two-dimensional array of pixel values that represents a digitized picture before it is transmitted over the network. The most widely-adopted standard relating to the compression of digitized pictures has been developed by an international standards body known as the **Joint Photographic Experts Group (JPEG)**. JPEG also forms the basis of most video compression algorithms and hence we shall limit our discussion of the compression of digitized pictures to describing the main principles of the JPEG standard.

3.4.5 JPEG

As we can deduce from the name, the JPEG standard was developed by a team of experts, each of whom had an in-depth knowledge of the compression of digitized pictures. They were working on behalf of the ISO, the ITU, and the IEC and JPEG is defined in the international standard **IS 10918**. In practice, the standard defines a range of different compression modes, each of which is intended for use in a particular application domain. We shall restrict our discussion here to the **lossy sequential mode** – also known as the **baseline mode** – since it is this which is intended for the compression of both monochromatic and color digitized pictures/images as used in multimedia communication applications. There are five main stages associated with this mode: image/block preparation, forward DCT, quantization, entropy encoding, and frame building. These are shown in Figure 3.14 and we shall discuss the role of each separately.

Image/block preparation

As we described in Section 2.4.3, in its pixel form, the source image/picture is made up of one or more 2-D matrices of values. In the case of a continuous-tone monochrome image, just a single 2-D matrix is required to store the set of 8-bit gray-level values that represent the image. Similarly, for a color image, if a CLUT is used just a single matrix of values is required.

Alternatively, if the image is represented in an *R*, *G*, *B* format three matrices are required, one each for the *R*, *G*, and *B* quantized values. Also, as we saw in Section 2.6.1 when we discussed the representation of a video signal, for color images the alternative form of representation known as Y, C_b, C_r can optionally be used. This is done to exploit the fact that the two chrominance signals, C_b and C_r, require half the bandwidth of the luminance signal, Y. This in turn allows the two matrices that contain the digitized chrominance components to be smaller in size than the Y matrix so producing a reduced form

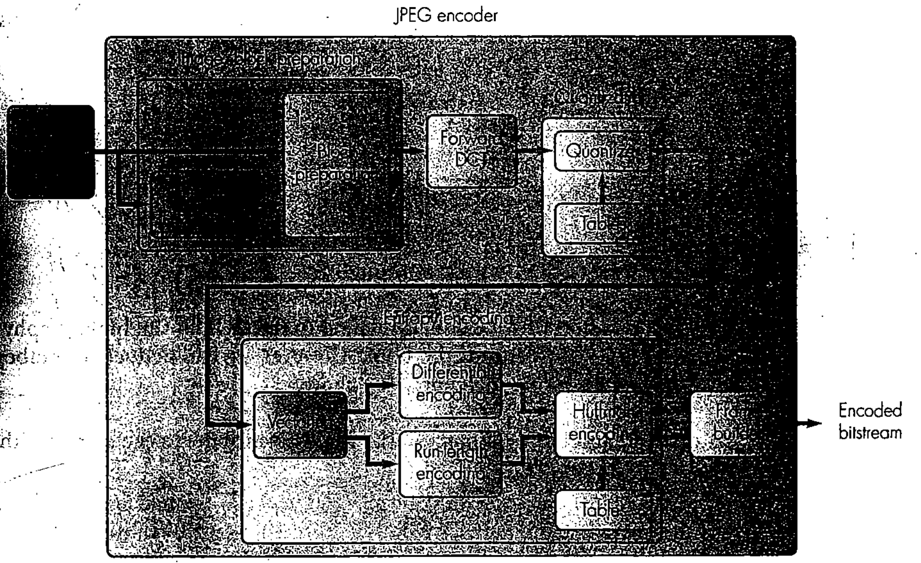


Figure 3.14 JPEG encoder schematic.

of representation over the equivalent R, G, B form of representation. For example, in the 4:2:0 format, groups of four neighboring chrominance values are averaged to produce a single value in the reduced matrix so reducing the size of the C_b and C_r matrices by a factor of four. The four alternative forms of representation are shown in Figure 3.15(a).

Once the source image format has been selected and prepared, the set of values in each matrix are compressed separately using the DCT. Before performing the DCT on each matrix, however, a second step known as **block preparation** is carried out. This is necessary since to compute the transformed value for each position in a matrix requires the values in all the locations of the matrix to be processed. It would be too time consuming to compute the DCT of the total matrix in a single step so each matrix is first divided into a set of smaller 8 × 8 submatrices. Each is known as a **block** and, as we can see in part (b) of the figure, these are then fed sequentially to the DCT which transforms each block separately.

Forward DCT

We described the principles of the DCT earlier in Section 3.2.4. Normally, each pixel value is quantized using 8 bits which produces a value in the range 0 to 255 for the intensity/luminance values – R, G, B, or Y – and a value in the range –128 to +127 for the two chrominance values – C_b and C_r. In order to compute the (forward) DCT, however, all the values are first centered around zero by subtracting 128 from each intensity/luminance value. Then, if the input 2-D matrix is represented by: $P[x, y]$ and the transformed matrix by $F[i, j]$, the DCT of each 8 × 8 block of values is computed using the expression:

$$F[i, j] = \frac{1}{4}C(i)\,C(j)\sum_{x=0}^{7}\sum_{y=0}^{7} P[x, y]\cos\frac{(2x+1)i\pi}{16}\cos\frac{(2y+1)j\pi}{16}$$

where $C(i)$ and $C(j) = 1/\sqrt{2}$ for $i, j = 0$

$= 1$ for all other values of i and j

and x, y, i and j all vary from 0 through 7.

You can find further details relating to the DCT in the bibliography for this chapter at the end of the book. However, we can deduce a number of points by considering the expression above:

- All 64 values in the input matrix, $P[x, y]$ contribute to each entry in the transformed matrix, $F[i, j]$.
- For $i = j = 0$, the two cosine terms (and hence horizontal and vertical frequency coefficients) are both 0. Also, since cos(0)=1, the value in location $F[0,0]$ of the transformed matrix is simply a function of the summation of all the values in the input matrix. Essentially, it is the mean of all 64 values in the matrix and is known as the **DC coefficient**.

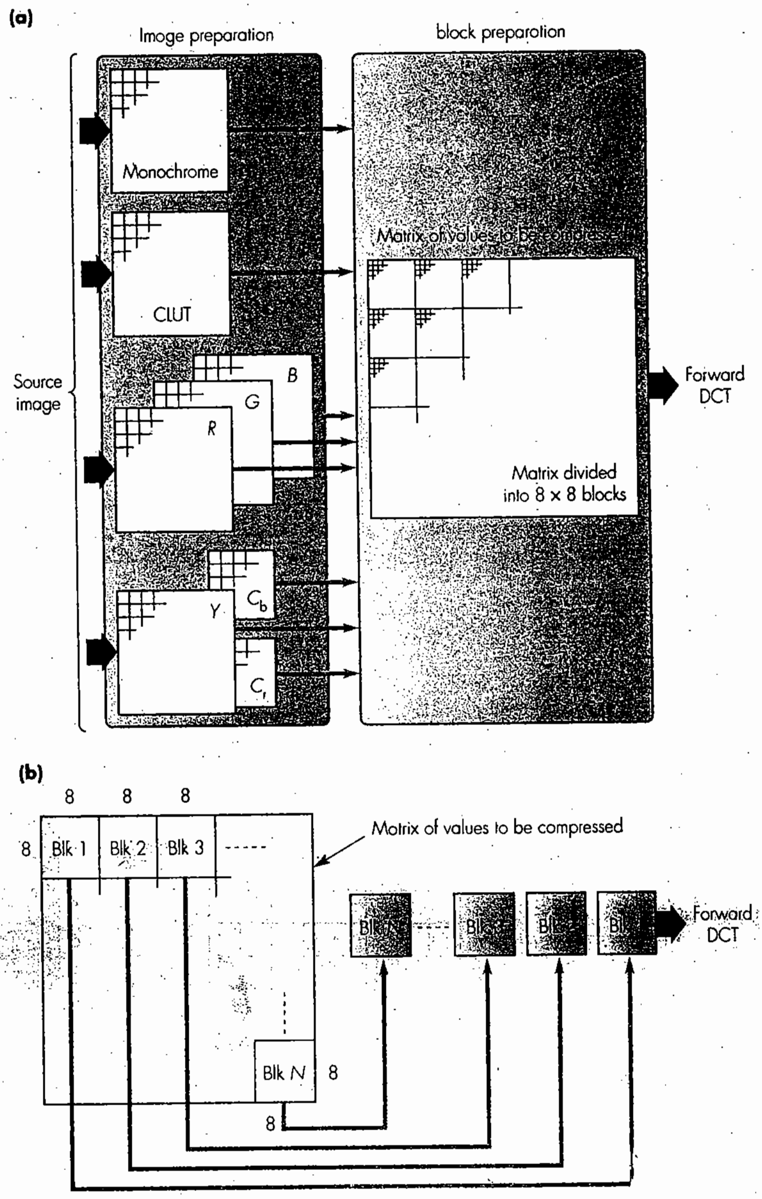
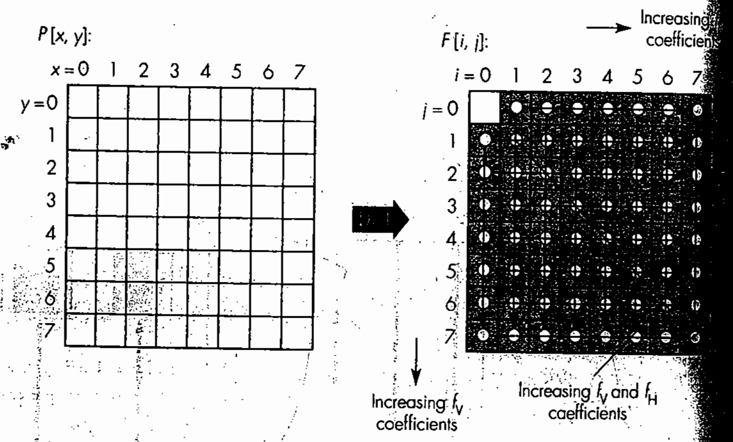


Figure 3.15 Image/block preparation: (a) image preparation; (b) block preparation.

- Since the values in all the other locations of the transformed matrix ha a frequency coefficient associated with them – either horizontal ($x=$ for $y=0$), vertical ($x=0$ for $y=1$–7) or both ($x=1$–7 for $y=1$-7) – the are known as **AC coefficients**.
- For $j=0$, only horizontal frequency coefficients are present which increase in frequency for $i=1$–7.
- For $i=0$, only vertical frequency coefficients are present which increa in frequency for $j=1$–7.
- In all other locations in the transformed matrix, both horizontal and vertical frequency coefficients are present to varying degrees.



$P[x, y]$ = 8 × 8 matrix of pixel values

$F[i, j]$ = 8 × 8 matrix of transformed values/spatial frequency coefficients

In $F[i, j]$: ☐ = DC coefficient ▨ = AC coefficients

f_H = horizontal spatial frequency coefficient

f_V = vertical spatial frequency coefficient

Figure 3.16 DCT computation features.

The above points are summarized in Figure 3.16. In order to gain a qu tative understanding of the likely values present in a transformed blo consider a typical image comprising, say, 640×480 pixels. Assuming a bl size of 8×8 pixels, the image will comprise 80×60 or 4800 blocks each which, for a screen width of, say, 16 inches (400mm), will occupy a square only 0.2×0.2 inches (5×5mm). Hence those regions of a picture that con a single color will generate a set of transformed blocks all of which will ha firstly, the same (or very similar) DC coefficient and secondly, only a few A coefficients within them. Thus it is only those areas of a picture which contain color transitions that will generate a set of transformed blocks with different DC coefficients and a larger number of AC coefficients within them. It is these features that are exploited in the quantization and entropy encoding phases of the compression algorithm.

Quantization

In theory, providing the forward DCT is computed to a high precision using, say, floating point arithmetic, there is very little loss of information during the DCT phase. Although in practice small losses occur owing to the use of fixed point arithmetic, the main source of information loss occurs during the quantization and entropy encoding stages where the compression takes place.

As we identified earlier in Section 3.2.4 when we first discussed transform encoding, the human eye responds primarily to the DC coefficient and the lower spatial frequency coefficients. Thus if the magnitude of a higher frequency coefficient is below a certain threshold, the eye will not detect it. This property is exploited in the quantization phase by dropping – in practice, setting to zero – those spatial frequency coefficients in the transformed matrix whose amplitudes are less than a defined threshold value. It should be noted, however, that although the eye is less sensitive to these frequency coefficients, once dropped, the same coefficients cannot be retrieved during the decoding procedure.

In addition to determining whether a particular spatial frequency coefficient is above a defined threshold, the quantization process aims to reduce the size of the DC and AC coefficients so that less bandwidth is required for their transmission. Instead of simply comparing each coefficient with the corresponding threshold value, a division operation is performed using the defined threshold value as the divisor. If the resulting (rounded) quotient is zero, the coefficient is less than the threshold value while if it is non-zero, this indicates the number of times the coefficient is greater than the threshold rather than its absolute value. For example, if the divisor is set to 16, then this will save 4 bits over the use of the absolute value. Clearly, this saving is at the expense of the precision used for the absolute values since in the decoder, these are determined by simply multiplying the received values by the corresponding threshold value.

As discussed, the sensitivity of the eye varies with spatial frequency, which implies that the amplitude threshold below which the eye will detect a particular spatial frequency also varies. In practice, therefore, the threshold values used vary for each of the 64 DCT coefficients. These are held in a two-dimensional matrix known as the **quantization table** with the threshold value to be used with a particular DCT coefficient in the corresponding position in the matrix.

Clearly, as we can see from the above, the choice of threshold values is important and, in practice, is a compromise between the level of compression that is required and the resulting amount of information loss that is accept-

ple 3.4

Assuming a quantization threshold value of 16, d
quantization error for each of the following DCT
coefficients:
127, 72, 64, 56, −56, −64, −72, −128.

Answer:

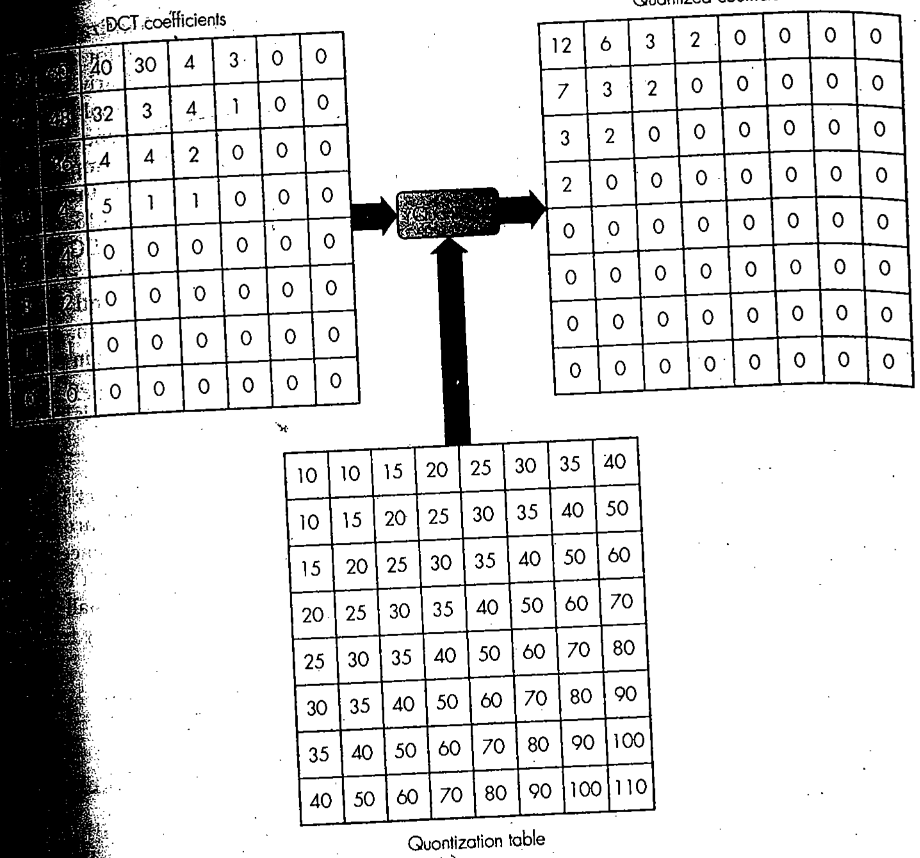
Coefficient	Quantized value	Rounded value	Dequantized value	Error
127	127/16 = 7.9375	8	[illegible]	[illegible]
72	4.5	5	[illegible]	[illegible]
64	4	4	[illegible]	[illegible]
56	3.5	4	[illegible]	[illegible]
−56	−3.5	−4	[illegible]	[illegible]
−64	−4	−4	[illegible]	[illegible]
−72	−4.5	−5	[illegible]	[illegible]
−128	−8	−8	[illegible]	[illegible]

As we can deduce from these figures, the maximum qua
is plus or minus 50% of the threshold value used.

able. Although the JPEG standard includes two default quantization tabl values – one for use with the luminance coefficients and the other for use wit the two sets of chrominance coefficients – it also allows for customized table to be used and sent with the compressed image. An example set of threshol values is given in the quantization table shown in Figure 3.17 together wit set of DCT coefficients and their corresponding quantized values. We can co clude a number of points from the values shown in the tables:

- The computation of the quantized coefficients involves rounding the quotients to the nearest integer value.
- The threshold values used, in general, increase in magnitude with increasing spatial frequency.
- The DC coefficient in the transformed matrix is largest.
- Many of the higher-frequency coefficients are zero.

It is the last two points that are exploited during the following entrop encoding stage.



DCT coefficients

[illegible]	[illegible]	40	30	4	3	0	0
[illegible]	[illegible]	32	3	4	1	0	0
[illegible]	[illegible]	4	4	2	0	0	0
[illegible]	[illegible]	5	1	1	0	0	0
[illegible]	[illegible]	0	0	0	0	0	0
[illegible]	[illegible]	0	0	0	0	0	0
[illegible]	[illegible]	0	0	0	0	0	0
[illegible]	0	0	0	0	0	0	0

Quantized coefficients

12	6	3	2	0	0	0	0
7	3	2	0	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

10	10	15	20	25	30	35	40
10	15	20	25	30	35	40	50
15	20	25	30	35	40	50	60
20	25	30	35	40	50	60	70
25	30	35	40	50	60	70	80
30	35	40	50	60	70	80	90
35	40	50	60	70	80	90	100
40	50	60	70	80	90	100	110

Quantization table

3.17 **Example computation of a set of quantized DCT coefficients.**

Entropy encoding

As we saw earlier in Figure 3.14, the entropy encoding stage comprises four steps: vectoring, differential encoding, run-length encoding, and Huffman encoding. We shall describe the role of each step separately.

Vectoring The various entropy encoding algorithms we described earlier in Section 3.2.3 operate on a one-dimensional string of values, that is, a vector. As we have just seen, however, the output of the quantization stage is a 2-D matrix of values. Hence before we can apply any entropy encoding to the set of values in the matrix, we must first represent the values in the form of a single-dimension vector. This operation is known as **vectoring**.

As we saw in Figure 3.17, the output of a typical quantization is matrix of values/coefficients which are mainly zeros except for a numb non-zero values in the top left-hand corner of the matrix. Clearly, if we si scanned the matrix using a line-by-line approach, then the resulting (1 vector would contain a mix of non-zero and zero values. In general, how this type of information structure does not lend itself to compressio order to exploit the presence of the large number of zeros in the quan matrix, a **zig-zag scan** of the matrix is used as shown in Figure 3.18.

As we can deduce from the figure, with this type of scan, the DC coeffi and lower-frequency AC coefficients – both horizontal and vertical – are sca first. Also, all the higher-frequency coefficients are in a sequential ord making this form of representation more suitable for compression. As we earlier in Figure 3.14, two different encoding schemes are applied in paral the values in the vector. The first is differential encoding, which is applied t DC coefficient only, and the second is run-length encoding, which is appli the remaining values in the vector containing the AC coefficients.

Differential encoding The first element in each transformed block is the coefficient which is a measure of the average color/luminance/chromin associated with the corresponding 8 × 8 block of pixel values. Hence it largest coefficient and, because of its importance, its resolution is kep high as possible during the quantization phase. Because of the small phy area covered by each block, the DC coefficient varies only slowly from block to the next.

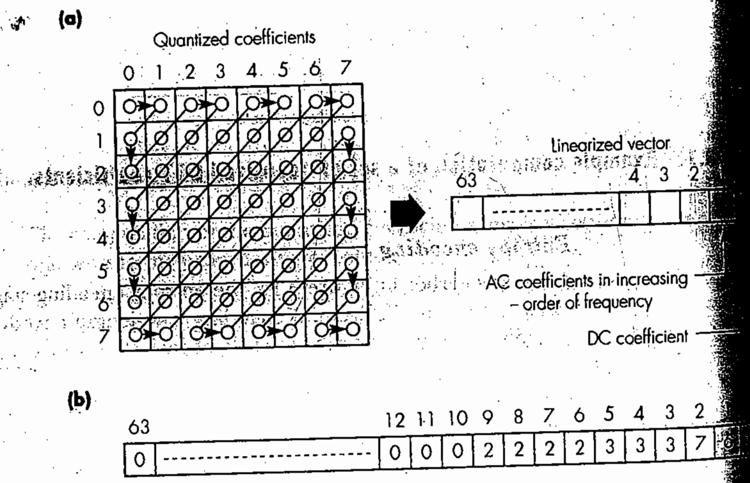


Figure 3.18 Vectoring using a zig-zag scan: (a) principle; (b) vecto example shown in Figure 3.17.

As we described in Section 3.2.4, the most efficient type of compression with this form of information structure is differential encoding since this encodes only the difference between each pair of values in a string rather than their absolute values. Hence in this application, only the difference in magnitude of the DC coefficient in a quantized block relative to the value in the preceding block is encoded. In this way, the number of bits required to encode the relatively large magnitudes of the DC coefficients is reduced.

For example, if the sequence of DC coefficients in consecutive quantized blocks – one per block – was:

12, 13, 11, 11, 10, ...

the corresponding difference values would be:

12, 1, −2, 0, −1, ...

the first difference value always being encoded relative to zero. The difference values are then encoded in the form (*SSS*, *value*) where the *SSS* field indicates the number of bits needed to encode the *value* and the value field the actual bits that represent the value. The rules used to encode each value are summarized in Figure 3.19(a).

As we can see, the number of bits required to encode each value is determined by its magnitude. A positive value is then encoded using the unsigned binary form and a negative value by the complement of this. Note also that a value of zero is encoded using a single 0 bit in the *SSS* field.

Example 3.5

[illegible]

Value	SSS	Value
[illegible]	[illegible]	[illegible]

(a)

Difference value	Number of bits needed (SSS)	Encoded value
0	0	
−1, 1	1	1 = 1 , −1 = 0
−3, −2, 2, 3	2	2 = 10 , −2 = 01 3 = 11 , −3 = 00
−7..−4, 4..7	3	4 = 100 , −4 = 011 5 = 101 , −5 = 010 6 = 110 , −6 = 001 7 = 111 , −7 = 000
−15...−8, 8...15	4	8 = 1000 , −8 = 0111 ⋮

(b)

Number of bits needed (SSS)	Huffman codeword
0	010
1	011
2	100
3	00
4	101
5	110
6	1110
7	11110
⋮	
11	111111110

Figure 3.19 Variable-length coding: (a) coding categories; (b) de Huffman codewords.

Run-length encoding The remaining 63 values in the vector are the A ficients and, because of the zig-zag scan, the vector contains long s zeros within it. To exploit this feature, the AC coefficients are encode form of a string of pairs of values. Each pair is made up of (*skip*, *valu* skip is the number of zeros in the run and *value* the next non-zer cient. Hence the 63 values in the vector shown earlier in Figure 3.18 w encoded as:

(0,6) (0,7) (0,3) (0,3) (0,3) (0,2) (0,2) (0,2) (0,2) (0,0)

Note that the final pair (0,0) indicates the end of the string for this blo that all the remaining coefficients in the block are zero. Also, that the field is encoded in the form *SSS*/*value*.

Example 3.6

Derive the binary form of the following run-length encoded AC coefficients:

(0,6) (0,7) (3,3) (0,−1) (0,0)

Answer:

AC coefficients	Skip	SSS/Value
0,6	0	3 110
0,7	0	3 111
3,3	3	2 11
0,−1	0	1 0
0,0	0	0

Huffman encoding As we saw in Section 3.4.3 when we described the encoding of digitized documents, significant levels of compression can be obtained by replacing long strings of binary digits by a string of much shorter codewords, the length of each codeword being a function of its relative frequency of occurrence. Normally, a table of codewords is used with the set of codewords precomputed using the Huffman coding algorithm. The same approach is used to encode the output of both the differential and run-length encoders.

For the differential-encoded DC coefficients in the block, the bits in the *SSS* field are not sent in their unsigned binary form as shown in Example 3.5 but in a Huffman-encoded form. This is done so that the bits in the *SSS* field have the prefix property – which we described earlier in Section 3.3.1 – and this enables the decoder to determine unambiguously the first *SSS* field from the received encoded bitstream.

Example 3.7

Determine the Huffman-encoded version of the following difference values which relate to the encoded DCT coefficients from consecutive DCT blocks:

12, 1, −2, 0, −1

Use for example purposes, the default Huffman codewords defined earlier in Figure 3.19(b).

3.7 Continued

Value	SSS	Huffman-encoded SSS	Encoded value	Encoded bitstream
12	4	101	1100	1011100
1	1	011	1	0111
-2	2	100	01	10001
0	0	010		010
-1	1	011	0	0110

As we can deduce from the set of Huffman-encoded *SSS* fields, they illustrate that providing the decoder uses the same set of codewords, it can readily determine the *SSS* field from the received (encoded) bitstream by searching the bitstream bit-by-bit – starting from the leftmost bit – until it reaches a valid codeword. The number of bits in the corresponding *SSS* value is then read from the table in Figure 3.19(b) and this is used to determine the number of following bits in the bitstream that contain the related *value*.

For each of the run-length encoded AC coefficients in the block, the bits that make up the *skip* and *SSS* fields are treated as a single (composite) symbol and this is then encoded using either the default table of Huffman codewords shown in Table 3.2 or a table of codewords that is sent with the encoded bitstream. Again, this is done so that the string of encoded composite symbols all have the prefix property so that the decoder can interpret the received bitstream on the correct coefficient boundaries. To enable the decoder to discriminate between the *skip* and *SSS* fields, each combination of the two fields is encoded separately and the composite symbol is then replaced by the equivalent Huffman codeword.

As we can deduce from Example 3.8, to decode the received bitstream the receiver first searches the bitstream – starting at the leftmost bit – for a valid codeword and, on finding this (100), determines the corresponding *skip* (0) and *SSS* (3) fields from the Huffman table. The *SSS* field is then used to determine the number of bits in the run-length value field and, after reading and decoding these, the process repeats until the EOB codeword is received indicating that the remaining coefficients are all zero. Because of the use of variable-length codewords in the various parts of the entropy encoding stage, this is also known as the **variable-length coding (VLC) stage**.

Table 3.2 Default Huffman codewords for encoding AC coefficients.

Skip/SSS	Codeword	Skip/SSS	Codeword	Skip/SSS	Codeword
[illegible]	[illegible]	3/8	1111111110010100	7/6	1111111110110010
[illegible]	[illegible]	3/9	1111111110010101	7/7	1111111110110011
[illegible]	[illegible]	3/10	1111111110010110	7/8	1111111110110100
[illegible]	[illegible]	4/1	111011	7/9	1111111110110101
[illegible]	[illegible]	4/2	1111111000	7/10	1111111110110110
[illegible]	[illegible]	4/3	1111111110010111	8/1	11111010
[illegible]	[illegible]	4/4	1111111110011000	8/2	111111111000000
[illegible]	[illegible]	4/5	1111111110011001	8/3	1111111110110111
[illegible]	[illegible]	4/6	1111111110011010	8/4	1111111110111000
[illegible]	[illegible]	4/7	1111111110011011	8/5	1111111110111001
[illegible]	[illegible]	4/8	1111111110011100	8/6	1111111110111010
[illegible]	[illegible]	4/9	1111111110011101	8/7	1111111110111011
[illegible]	[illegible]	4/10	1111111110011110	8/8	1111111110111100
[illegible]	[illegible]	5/1	1111010	8/9	1111111110111101
[illegible]	[illegible]	5/2	1111111001	8/10	1111111110111110
[illegible]	[illegible]	5/3	1111111110011111	9/1	111111000
[illegible]	[illegible]	5/4	1111111110100000	9/2	1111111110111111
[illegible]	[illegible]	5/5	1111111110100001	9/3	1111111111000000
[illegible]	[illegible]	5/6	1111111110100010	9/4	1111111111000001
[illegible]	[illegible]	5/7	1111111110100011	9/5	1111111111000010
[illegible]	[illegible]	5/8	1111111110100100	9/6	1111111111000011
[illegible]	[illegible]	5/9	1111111110100101	9/7	1111111111000100
[illegible]	[illegible]	5/10	1111111110100110	9/8	1111111111000101
[illegible]	[illegible]	6/1	1111011	9/9	1111111111000110
[illegible]	[illegible]	6/2	11111111000	9/10	1111111111000111
[illegible]	[illegible]	6/3	1111111110100111	10/1	111111001
[illegible]	[illegible]	6/4	1111111110101000	10/2	1111111111001000
[illegible]	[illegible]	6/5	1111111110101001	10/3	1111111111001001
[illegible]	[illegible]	6/6	1111111110101010	10/4	1111111111001010
[illegible]	[illegible]	6/7	1111111110101011	10/5	1111111111001011
[illegible]	[illegible]	6/8	1111111110101100	10/6	1111111111001100
[illegible]	[illegible]	6/9	1111111110101101	10/7	1111111111001101
[illegible]	[illegible]	6/10	1111111110101110	10/8	1111111111001110
[illegible]	[illegible]	7/1	11111001	10/9	1111111111001111
[illegible]	1111111110010000	7/2	11111111001	10/10	1111111111010000
[illegible]	1111111110010001	7/3	1111111110101111	11/1	111111010
3/6	1111111110010010	7/4	1111111110110000	11/2	1111111111010001
3/7	1111111110010011	7/5	1111111110110001	11/3	1111111111010010

▶

.2 Continued

SS	Codeword	Skip/SSS	Codeword	Skip/SSS	Codeword
	1111111111010011	13/6	1111111111100111	15/7	[illegible]
	1111111111010100	13/7	1111111111101000	15/8	[illegible]
	1111111111010101	13/8	1111111111101001	15/9	[illegible]
	1111111111010110	13/9	1111111111101010	15/10	[illegible]
	1111111111010111	13/10	1111111111101011		
	1111111111011000	14/1	111111110110		
	1111111111011001	14/2	1111111111101100		
	1111111010	14/3	1111111111101101		
	1111111111011010	14/4	1111111111101110		
	1111111111011011	14/5	1111111111101111		
	1111111111011100	14/6	1111111111110000		
	1111111111011101	14/7	1111111111110001		
	1111111111011110	14/8	1111111111110010		
	1111111111011111	14/9	1111111111110011		
	1111111111100000	14/10	1111111111110100		
	1111111111100001	15/0	11111111001		
	1111111111100010	15/1	1111111111110101		
	11111111010	15/2	1111111111110110		
	1111111111100011	15/3	1111111111110111		
	1111111111100100	15/4	1111111111111000		
	1111111111100101	15/5	1111111111111001		
	1111111111100110	15/6	1111111111111010		

end of block

xample 3.8

Derive the composite binary symbols for the following [illegible] encoded AC coefficients:

(0,6)(0,7)(3,3)(0,−1)(0,0)

Assuming the *skip* and SSS fields are both encoded as a [illegible] symbol, use the Huffman codewords shown in Table 3.2 [illegible] Huffman-encoded bitstream for this set of symbols.

3.8 Continued

Answer:

The *skip* and SSS fields for this set of AC coefficients were derived earlier in Example 3.6. Hence:

AC coefficient	Composite symbol skip	SSS	Huffman codeword	Run-length value
0,6	0	3	100	6 = 110
0,7	0	3	100	7 = 111
3,3	3	2	111110111	2 = 10
0,−1	0	1	00	1 = 0
0,0	0	0	1010	

The Huffman-encoded bitstream is then derived by adding the run-length encoded value to each of the Huffman codewords:

100110 100111 11111011110 000 1010

In this example, the number of bits required to transmit the set of AC coefficients for the 8 × 8 block of pixels is 29 and, assuming 6 bits for the DC coefficient, the total for the block is 35 bits. Hence assuming each pixel value is 8 bits, the resulting compression ratio for this block is 512/35 or, approximately, 14.6:1.

Frame building

Typically, the bitstream output by a JPEG encoder – corresponding to, say, the compressed version of a printed picture – is stored in the memory of a computer ready for either integrating with other media if necessary or accessing from a remote computer. As we can see from the above, in order for the decoder in the remote computer to be able to interpret all the different fields and tables that make up the bitstream, it is necessary to delimit each field and set of table values in a defined way. The JPEG standard, therefore, also includes a definition of the structure of the total bitstream relating to a particular image/picture. This is known as a *frame* and its outline structure is shown in Figure 3.20.

The role of the **frame builder** shown earlier in Figure 3.14 is to encapsulate all the information relating to an encoded image/picture in this format and, as we can see, the structure of a frame is hierarchical. At the top level, the complete frame-plus-header is encapsulated between a *start-of-frame* and an *end-of-frame* delimiter which allows the receiver to determine the start and

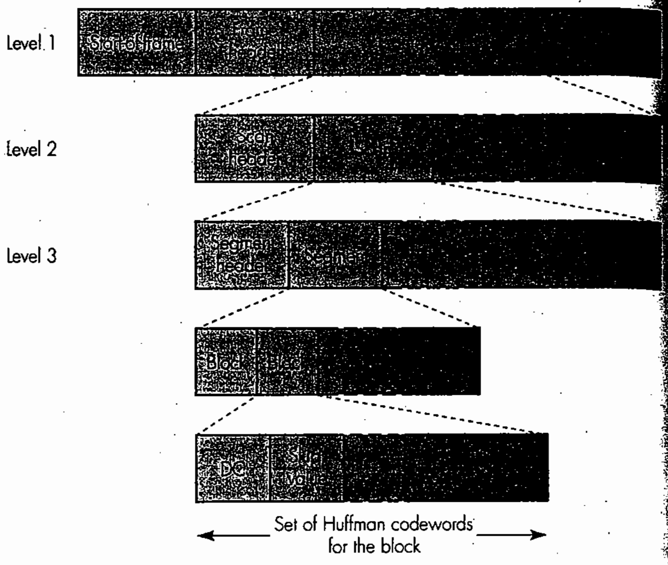


Figure 3.20 JPEG encoder output bitstream format.

end of all the information relating to a complete image/picture. The *fr* *header* contains a number of fields that include:

- the overall width and height of the image in pixels;
- the number and type of components that are used to represent the image (CLUT, R/G/B, $Y/C_b/C_r$);
- the digitization format used (4:2:2, 4:2:0 etc.).

At the second level, a frame consists of a number of components eac which is known as a *scan*. These are also preceded by a header which con fields that include:

- the identity of the components (R/G/B etc.);
- the number of bits used to digitize each component;
- the quantization table of values that have been used to encode each component.

Typically, each scan/component comprises one or more *segments* each which can contain a group of (8 × 8) blocks preceded by a header. This co tains the Huffman table of values that have been used to encode each blo

in the segment should the default tables not be used. In this way, each segment can be decoded independently of the others which overcomes the possibility of bit errors propagating and affecting other segments. Hence each complete frame contains all the information necessary to enable the JPEG decoder to identify each field in a received frame and then perform the corresponding decoding operation.

JPEG decoding

As we can see in Figure 3.21, a JPEG decoder is made up of a number of stages which are simply the corresponding decoder sections of those used in the encoder. Hence the time to carry out the decoding function is similar to that used to perform the encoding.

On receipt of the encoded bitstream the **frame decoder** first identifies the control information and tables within the various headers. It then loads the contents of each table into the related table and passes the control information to the **image builder**. It then starts to pass the compressed bitstream to the Huffman decoder which carries out the corresponding decompression operation using either the default or the preloaded table of codewords. The two decompressed streams containing the DC and AC coefficients of each block are then passed to the differential and run-length decoders respectively. The resulting matrix of values is then dequantized using either the default or the preloaded values in the quantization table.

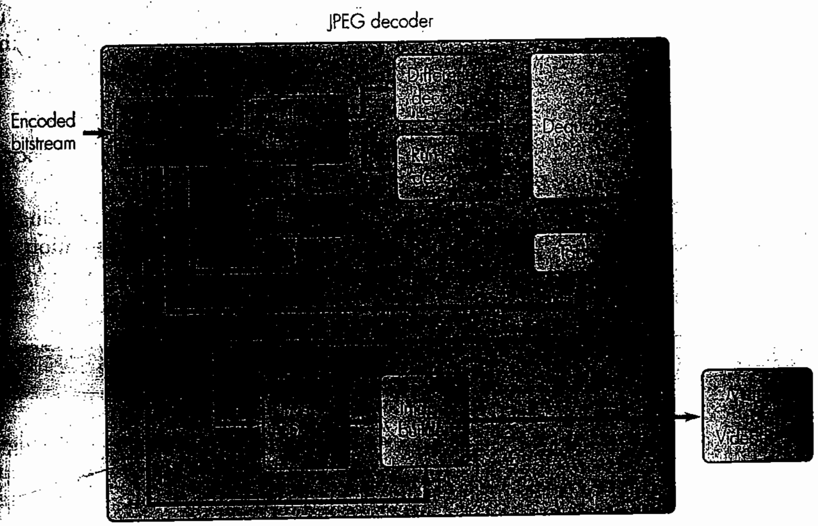


Figure 3.21 JPEG decoder schematic.

Each resulting block of 8 × 8 spatial frequency coefficients is passe turn to the **inverse DCT** which transforms them back into their spatial fo using the expression:

$$P[x, y] = \frac{1}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} C(i)\, C(j)\, F[i, j] \cos \frac{(2x+1)i\pi}{16} \cos \frac{(2y+1)j\pi}{16}$$

where $C(i)$ and $C(j) = 1/\sqrt{2}$ for $i, j = 0$
$= 1$ for all other values of i and j.

The image builder then reconstructs the original image from th blocks using the control information passed to it by the frame decod Although the JPEG standard is relatively complicated owing to the numbe encoding/decoding stages associated with it, compression ratios in exces 20:1 can be obtained while still retaining a good quality output image/p ture. This level of compression, however, applies to pictures whose conten relatively simple – that is, have relatively few color transitions – and, for mo complicated pictures, compression ratios nearer to 10:1 are more commo These figures, however, assume each pixel location has three planes asso ated with it – R/G/B or $Y/C_b/C_r$ – and hence if a CLUT is used, then bo figures can be multiplied by a factor of 3. Nevertheless, even with a compr sion ratio of 10:1, the amount of memory required with the various types display tabulated in Table 2.1 is reduced to a range of from 30 kbytes throu to 240 kbytes. More importantly, the time delay incurred in accessing th images is reduced by a factor of 10.

Finally, as with the GIF, it is also possible to encode and rebuild the ima in a progressive way by first sending an outline of the image and then p gressively adding more detail to it. This can be achieved in the following w

- **progressive mode:** in this mode, first the DC and low-frequency coefficients of each block are sent and then the higher-frequency coefficients;
- **hierarchial mode:** in this mode, the total image is first sent using a lo resolution – for example 320×240 – then at a higher resolution such 640×480.

Summary

In this chapter we have described a selection of the compression algorith that are used for the compression of text and images. In general, compr sion is applied to both media types in order to reduce the time taken transfer the source information over a network. This is done either to redu the cost of the network connection or, in interactive applications, to redu the response time to a request for the source information.

The basic techniques associated with all compression algorithms were first identified and described. These were classified as being either entropy encoding algorithms or source encoding. Entropy encoding exploits how the source information is represented and we described two examples: run-length encoding and statistical encoding.

Run-length encoding is used when the source information contains long strings of the same symbol such as a character, a bit, or a byte. Instead of sending the source information in the form of independent codewords, it is sent by simply indicating the particular symbol in each string together with an indication of the number of symbols in the string.

Statistical encoding exploits the fact that not all symbols in the source information occur with equal probability. Hence, instead of encoding all the symbols with fixed-length codewords, variable-length codewords are used with the shortest ones used to encode those symbols that occur most frequently.

In contrast, source encoding exploits a particular property of the source information in order to produce an alternative form of representation that is either a compressed version of the original form or is more amenable to the application of compression. Two examples were described: differential encoding and transform encoding.

Differential encoding is used when the amplitude of the values that make up the source information cover a large range but the difference between successive values is relatively small. Instead of using a set of relatively large codewords to represent the actual amplitude of each value, a set of smaller codewords is used, each of which indicates only the difference in amplitude between the current value being encoded and the immediately preceding value.

As the name implies, transform encoding involves transforming the source information into an alternative form of representation that lends itself more readily to the application of compression. The example that we described was the discrete cosine transform (DCT). This is used for image compression and transforms the matrix of pixel values that represent the image into a matrix of spatial frequency components which, in turn, lends itself more readily to the application of compression.

In terms of text compression, when the compressed source information is decompressed by the receiver, there is normally no loss of information. The compression algorithms that have this property are known as lossless and we described a number of such algorithms. These included both static and dynamic Huffman coding, arithmetic coding, and the LZW coding algorithm. The two Huffman algorithms and arithmetic coding are based on the relative frequency of occurrence of single characters in the source information and the LZW algorithm strings of characters.

In terms of image compression, we described a number of the algorithms that are used for the compression of graphical images, digitized documents, and digitized pictures. For use with graphical images we described the GIF and TIFF standards, for digitized documents two modified Huffman coding algorithms, and for digitized pictures the JPEG algorithm. All the algorithms that were described are part of international standards which, in addition to the compression algorithm, also define the format of the compressed information when it is being stored or transferred across a network.

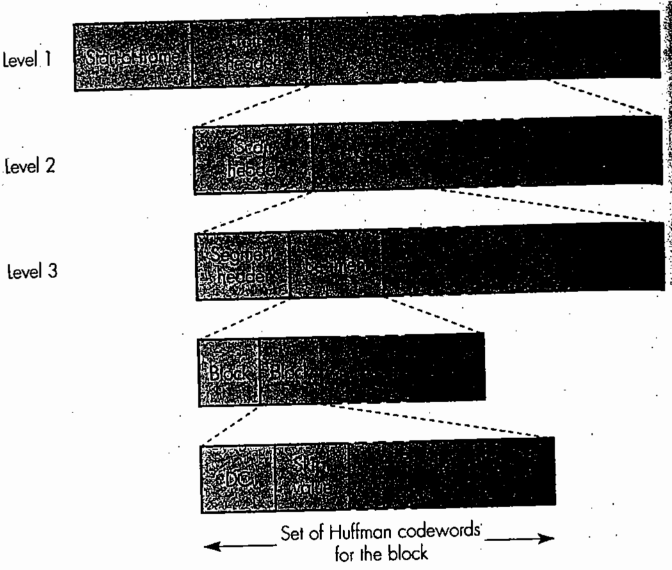


Figure 3.20 JPEG encoder output bitstream format.

end of all the information relating to a complete image/picture. The *fra*
header contains a number of fields that include:

- the overall width and height of the image in pixels;
- the number and type of components that are used to represent the image (CLUT, R/G/B, $Y/C_b/C_r$);
- the digitization format used (4:2:2, 4:2:0 etc.).

At the second level, a frame consists of a number of components eac
which is known as a *scan*. These are also preceded by a header which con
fields that include:

- the identity of the components (R/G/B etc.);
- the number of bits used to digitize each component;
- the quantization table of values that have been used to encode each component.

Typically, each scan/component comprises one or more *segments* eac
which can contain a group of (8 × 8) blocks preceded by a header. This co
tains the Huffman table of values that have been used to encode each blo

in the segment should the default tables not be used. In this way, each segment can be decoded independently of the others which overcomes the possibility of bit errors propagating and affecting other segments. Hence each complete frame contains all the information necessary to enable the JPEG decoder to identify each field in a received frame and then perform the corresponding decoding operation.

JPEG decoding

As we can see in Figure 3.21, a JPEG decoder is made up of a number of stages which are simply the corresponding decoder sections of those used in the encoder. Hence the time to carry out the decoding function is similar to that used to perform the encoding.

On receipt of the encoded bitstream the **frame decoder** first identifies the control information and tables within the various headers. It then loads the contents of each table into the related table and passes the control information to the **image builder**. It then starts to pass the compressed bitstream to the Huffman decoder which carries out the corresponding decompression operation using either the default or the preloaded table of codewords. The two decompressed streams containing the DC and AC coefficients of each block are then passed to the differential and run-length decoders respectively. The resulting matrix of values is then dequantized using either the default or the preloaded values in the quantization table.

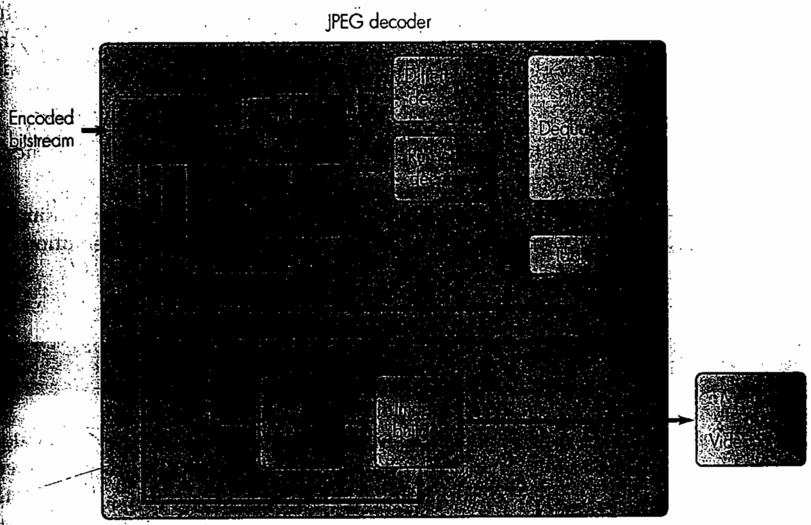


Figure 3.21 JPEG decoder schematic.

Each resulting block of 8 × 8 spatial frequency coefficients is passed turn to the **inverse DCT** which transforms them back into their spatial fo using the expression:

$$P[x,y] = \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} C(i)\,C(j)\,F[i,j]\cos\frac{(2x+1)i\pi}{16}\cos\frac{(2y+1)j\pi}{16}$$

where $C(i)$ and $C(j) = 1/\sqrt{2}$ for $i, j = 0$
$= 1$ for all other values of i and j.

The image builder then reconstructs the original image from the blocks using the control information passed to it by the frame decode Although the JPEG standard is relatively complicated owing to the number encoding/decoding stages associated with it, compression ratios in excess 20:1 can be obtained while still retaining a good quality output image/pi ture. This level of compression, however, applies to pictures whose content relatively simple – that is, have relatively few color transitions – and, for mo complicated pictures, compression ratios nearer to 10:1 are more commo These figures, however, assume each pixel location has three planes asso ated with it – R/G/B or $Y/C_b/C_r$ – and hence if a CLUT is used, then bo figures can be multiplied by a factor of 3. Nevertheless, even with a compre sion ratio of 10:1, the amount of memory required with the various types display tabulated in Table 2.1 is reduced to a range of from 30 kbytes throug to 240 kbytes. More importantly, the time delay incurred in accessing the images is reduced by a factor of 10.

Finally, as with the GIF, it is also possible to encode and rebuild the imag in a progressive way by first sending an outline of the image and then pr gressively adding more detail to it. This can be achieved in the following wa

- **progressive mode:** in this mode, first the DC and low-frequency coefficients of each block are sent and then the higher-frequency coefficients;
- **hierarchial mode:** in this mode, the total image is first sent using a low resolution – for example 320×240 – then at a higher resolution such 640×480.

Summary

In this chapter we have described a selection of the compression algorith that are used for the compression of text and images. In general, compre sion is applied to both media types in order to reduce the time taken t transfer the source information over a network. This is done either to reduc the cost of the network connection or, in interactive applications, to reduc the response time to a request for the source information.

The basic techniques associated with all compression algorithms were first identified and described. These were classified as being either entropy encoding algorithms or source encoding. Entropy encoding exploits how the source information is represented and we described two examples: run-length encoding and statistical encoding.

Run-length encoding is used when the source information contains long strings of the same symbol such as a character, a bit, or a byte. Instead of sending the source information in the form of independent codewords, it is sent by simply indicating the particular symbol in each string together with an indication of the number of symbols in the string.

Statistical encoding exploits the fact that not all symbols in the source information occur with equal probability. Hence, instead of encoding all the symbols with fixed-length codewords, variable-length codewords are used with the shortest ones used to encode those symbols that occur most frequently.

In contrast, source encoding exploits a particular property of the source information in order to produce an alternative form of representation that is either a compressed version of the original form or is more amenable to the application of compression. Two examples were described: differential encoding and transform encoding.

Differential encoding is used when the amplitude of the values that make up the source information cover a large range but the difference between successive values is relatively small. Instead of using a set of relatively large codewords to represent the actual amplitude of each value, a set of smaller codewords is used, each of which indicates only the difference in amplitude between the current value being encoded and the immediately preceding value.

As the name implies, transform encoding involves transforming the source information into an alternative form of representation that lends itself more readily to the application of compression. The example that we described was the discrete cosine transform (DCT). This is used for image compression and transforms the matrix of pixel values that represent the image into a matrix of spatial frequency components which, in turn, lends itself more readily to the application of compression.

In terms of text compression, when the compressed source information is decompressed by the receiver, there is normally no loss of information. The compression algorithms that have this property are known as lossless and we described a number of such algorithms. These included both static and dynamic Huffman coding, arithmetic coding, and the LZW coding algorithm. The two Huffman algorithms and arithmetic coding are based on the relative frequency of occurrence of single characters in the source information and the LZW algorithm strings of characters.

In terms of image compression, we described a number of the algorithms that are used for the compression of graphical images, digitized documents, and digitized pictures. For use with graphical images we described the GIF and TIFF standards, for digitized documents two modified Huffman coding algorithms, and for digitized pictures the JPEG algorithm. All the algorithms that were described are part of international standards which, in addition to the compression algorithm, also define the format of the compressed information when it is being stored or transferred across a network.

Exercises

n 3.2

Explain the meaning of the following terms elating to compression:
- (i) source encoders and destination decoders,
- (ii) lossless and lossy compression,
- (iii) entropy encoding,
- (iv) source encoding.

Explain the meaning of the following terms elating to statistical encoding:
- (i) run-length encoding,
- (ii) statistical encoding.

Explain the meaning of the following terms elating to statistical encoding:
- (i) prefix property,
- (ii) entropy,
- (iii) Shannon's formula,
- (iv) coding efficiency.

Explain the meaning of the following terms relating to source encoding:
- (i) differential encoding,
- (ii) transform encoding

With the aid of diagrams, explain in a qualitative way the meaning of the following terms relating to transform encoding:
- (i) spatial frequency,
- (ii) horizontal and vertical frequency components,
- (iii) discrete cosine transform (DCT).

on 3.3

Explain the meaning of the following terms relating to text compression algorithms:
- (i) static coding,
- (ii) dynamic/adaptive coding.

With the aid of an example, describe the rules that are followed to construct the Huffman code tree for a transmitted character set.

Messages comprising seven different characters, A through G, are to be transmitted over a data link. Analysis has shown that the relative frequency of occurrence of each character is:

A 0.10, B 0.25, C 0.05, D 0.32, E 0.01, F 0.07, G 0.2

- (i) Derive the entropy of the messages.
- (ii) Use static Huffman coding to derive a suitable set of codewords.
- (iii) Derive the average number of bits per codeword for your codeword set to transmit a message and compare this with both the fixed-length binary and ASCII codewords.

3.9 (i) State the prefix property of Huffman codes and hence show that your codeword set derived in Exercise 3.8 satisfied this.
- (ii) Derive a flowchart for an algorithm to decode a received bit string encoded using your codeword set.
- (iii) Give an example of the decoding operation assuming the received bit string comprises a mix of the seven characters.

3.10 The following character string is to be transmitted using Huffman coding:

ABACADABACADABACABAB

- (i) Derive the Huffman code tree.
- (ii) Determine the savings in transmission bandwidth over normal ASCII and binary coding.

3.11 With reference to the example shown in Figure 3.6 relating to dynamic Huffman coding:
- (i) Write down the actual transmitted bit pattern corresponding to the character string:

  "This is"

  assuming ASCII coding is being used.
- (ii) Deduce the extensions to the existing Huffman tree if the next word transmitted is "the".

3.12 Assuming the character set and associate probability assignments given in Figure 3.7, derive the codeword value for the character string *newt.* Assuming this is recieved by the destination, explain how the decoder determines the original string from the received codeword value.

3.13 Explain the principle of operation of the LZ compression algorithm. Hence assuming a dictionary of 16 000 words and an average word length of 5 bits, derive the average compression ratio that is achieved relative to using 7-bit ASCII codewords.

3.14 Explain the principle of operation of the LZW compression algorithm and how this is different from the LZ algorithm.

3.15 Assume the contents of a file that consists of 256 different words – each composed of alphanumeric characters from the basic ASCII character set – is to be sent over a network using the LZW algorithm. If the file contents start with the string:

This is easy as it is easy ...

show the entries in the dictionary of the encoder up to this point and the string of codewords that are sent. Also show how the receiver builds up its own dictionary and determines the original file contents from this.

3.16 Assume the same message as in Exercise 3.15 but with the number of different words much larger and unknown. How can the algorithm be changed to accommodate this?

Section 3.4

3.17 Explain the basic mode of operation of GIF. Include in your explanation the size of the color table used, how each pixel value is sent, and how the receiver knows the image parameters used by this source.

3.18 In relation to GIF, explain how the LZW coding algorithm can be applied to the (compressed) image data. Include in your explanation how compression is achieved and how the receiver interprets the compressed information.

3.19 With the aid of a diagram, describe the interlaced mode of operation of GIF. Include the potential applications of this mode and how the receiver knows it is being used.

3.20 Describe the principles of TIFF and its application domains.

3.21 Explain the meaning of the following terms relating to facsimile machines:
- (i) termination codes,
- (ii) make-up codes,
- (iii) overscanning,
- (iv) the EOL code and its uses.

3.22 Discriminate between a one-dimensional coding scheme and a two-dimensional (MMR) scheme.

3.23 Given a scanned line of pels, assuming a one-dimensional coding scheme, deduce an algorithm
- (i) to determine the transmitted codewords, and
- (ii) to decode the received string of codewords. Use the Huffman tables in Fig. 3.11 as a guide.

3.24 With the aid of pel patterns, assuming an MMR coding scheme, explain the meaning of the following terms:
- (i) pass mode,
- (ii) vertical mode,
- (iii) horizontal mode. Hence with the aid of the code table given in Table 3.1, deduce an algorithm to perform the encoding operation.

3.25 With the aid of a diagram, identify the five main stages associated with the baseline mode of operation of JPEG and give a brief description of the role of each stage.

3.26 With the aid of a diagram, explain how the individual 8×8 blocks of pixel values are derived by the image and block preparation stage for each of the following source image forms:
- (i) monochrome/CLUT,
- (ii) RGB,
- (iii) Y, C_b, C_r.

In relation to the Y, C_b, C_r format, show the order in which the blocks are output.

h the aid of a diagram, explain the mean-
of the following terms relating to the DCT
orithm:
DC coefficient,
horizontal and vertical spatial frequency
coefficients.

nce by considering a typical image of 1024 ×
8 pixels displayed on a 17 inch (432 mm)
een, explain where the savings in bandwidth
ise with JPEG.

ate the characteristic of the eye that is
ploited in the quantization phase of the
EG algorithm. Hence assuming the set of DC
efficients and threshold values shown in
igure 3.17, explain how the set of quantiza-
on coefficients are derived.

Jse a range of DCT coefficients and a selected
quantization threshold to derive how the maxi-
num quantization error is determined by the
choice of threshold value.

State the characteristics of the values in the quantized coefficient matrix that are exploited during the entropy encoding stage. Why is vectoring using a zig-zag scan applied to the matrix?

Explain why differential encoding is used for the compression of the DC coefficients in successive blocks. By means of an example set of coefficients, estimate the savings in bandwidth that are achieved.

Using the set of coding categories listed in Figure 3.19(a), determine the encoded version of the following string of DC coefficients.

16, 15, 16, 14, 12, ...

3.33 Describe how the 63 quantized AC coeffi
in a vector are encoded using run-length en
ing. Hence derive the encoded form of
following vector of quantized AC coefficien

6, 7, 0, 0, 0, 3, –1, 0, 0, ..., 0.

3.34 Describe how the differential-encoded
coefficients from a string of successive blo
are set using Huffman encoding. Hence ass
ing the default Huffman codewords show
Figure 3.19(b), derive the encoded bitstr
for the set of differential encoded DC co
cients you derived in Exercise 3.32.

3.35 Describe how the set of run-length encoded
coefficients for a block are sent using Huffm
encoding. Hence assuming the set of def
Huffman codewords listed in Table 3.2, de
the Huffman-encoded bitstream for the se
run-length encoded coefficients you derived
Exercise 3.33. How is the end of the se
encoded coefficients for a block determined

3.36 All the information relating to a compres
image/picture generated by the various sta
in the JPEG encoder is encapsulated with
single frame in such a way that the decoder
identify the individual fields that are pres
Show the structure of a frame in a diagram
describe the role of the main fields in each
the headers that are used.

3.37 With the aid of Figure 3.21, explain how
various parts of the encoded frame iden
in Exercise 3.36 are used to recreate the o
nal image.
Identify the two alternate ways that ca
used to recreate an image in a progressive

4

Audio and video compression

4.1 Introduction

As we described in Chapter 2, unlike text and images, both audio and most video signals are continuously varying analog signals. Hence, as we saw in Section 2.2, after digitization, both comprise a continuous stream of digital values, each value representing the amplitude of a sample of the particular analog signal taken at repetitive time intervals. As a result, the compression algorithms associated with digitized audio and video are different from those we described in the last chapter. In this chapter, therefore, we shall describe a selection of these algorithms, first those relating to audio and then those relating to video.

4.2 Audio compression

We discussed the digitization of audio signals in Section 2.5 and, as we saw, the digitization process is known as **pulse code modulation** or **PCM**. This involves sampling the (analog) audio signal/waveform at a minimum rate which is twice that of the maximum frequency component that makes up the signal. Alternatively, if the (frequency) bandwidth of the communicatio channel to be used is less than that of the signal, then the sampling rat determined by the bandwidth of the communications channel. The latte then known as a **bandlimited signal**.

In the case of a speech signal, the maximum frequency componen 10 kHz and hence the minimum sampling rate is 20 ksps. For general audi music for example, the figures are 20 kHz and 40 ksps respectively. Th number of bits per sample also varies and, typically, is 12 bits for speech an 16 bits for general audio. In addition, for a stereophonic signal, two signal must be digitized. As we saw in Example 2.4, this produces a bit rate o 240 kbps for a speech signal and 1.28 Mbps for a stereophonic music/genera audio signal, the latter being the rate used with compact discs.

In practice, however, in most multimedia applications involving audio the bandwidth of the communication channels that are available dictate rate that are less than these. This can be achieved in one of two ways: either th audio signal is sampled at a lower rate (and, if necessary, with fewer bits pe sample) or a compression algorithm is used. Although the first approach i relatively simple to implement, using a lower sampling rate has the disadvantage that the quality of the decoded signal is reduced owing to the loss of the higher-frequency components from the original signal. Also, the use of fewer bits per sample results in the introduction of higher levels of quantization noise. Normally, therefore, a compression algorithm is used since, as we shall see, this achieves a comparable perceptual quality (as perceived by the ear) to that obtained with a higher sampling rate but with a reduced bandwidth requirement. In this section we consider a selection of the compression algorithms that are widely used for audio in a range of multimedia applications.

4.2.1 Differential pulse code modulation

Differential pulse code modulation (**DPCM**) is a derivative of standard PCM and exploits the fact that, for most audio signals, the range of the differences in amplitude between successive samples of the audio waveform is less than the range of the actual sample amplitudes. Hence if only the digitized difference signal is used to encode the waveform then fewer bits are required than for a comparable PCM signal with the same sampling rate. A DPCM encoder and decoder are shown in Figure 4.1(a) and a simplified timing diagram for the encoder is shown in Figure 4.1(b).

Essentially, the previous digitized sample of the analog input signal is held in the register (R) – a temporary storage facility – and the difference signal (DPCM) is computed by subtracting the current contents of the register (R_0) from the new digitized sample output by the ADC (PCM). The value in the register is then updated by adding to the current register contents the computed difference signal output by the subtractor prior to its transmission. The decoder operates by simply adding the received difference signal (DPCM) to the previously computed signal held in the register (PCM). Typical savings with DPCM are limited to just 1 bit which, for a standard PCM voice signal, for example, reduces the bit rate requirement from 64 kbps to 56 kbps.

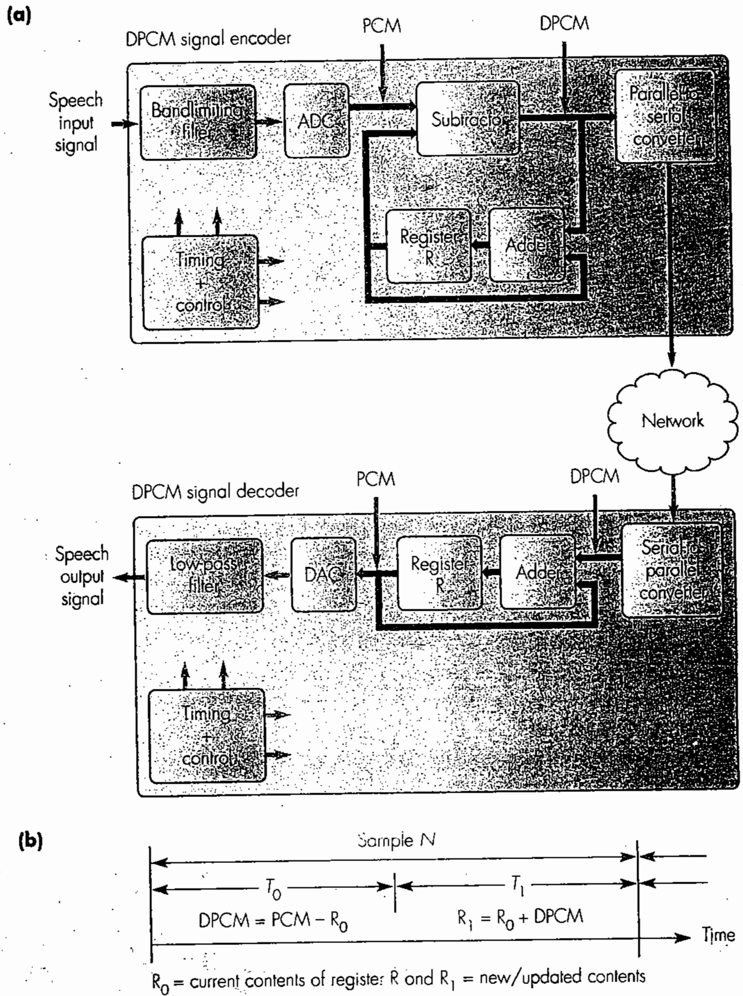


Figure 4.1 DPCM principles: (a) encoder/decoder schematic; (b) encoder timing.

As we can deduce from the circuit shown in Figure 4.1(a), the output of the ADC is used directly and hence the accuracy of each computed difference signal – also known as the **residual** (signal) – is determined by the accuracy of the previous signal/value held in the register. As we saw in Section 2.2, all ADC operations produce quantization errors and hence a string of, say, positive errors, will have a cumulative effect on the accuracy of the value that is held in the register. This means, therefore, that with a ba DPCM scheme, the previous value held in the register is only an approxi tion. Hence more sophisticated techniques have been developed f estimating – also known as predicting – a more accurate version of the pre ous signal. To achieve this, these predict the previous signal by using not o the estimate of the current signal but also varying proportions of a number the immediately preceding estimated signals. The proportions used are dete mined by what are known as **predictor coefficients** and the principle is show in Figure 4.2.

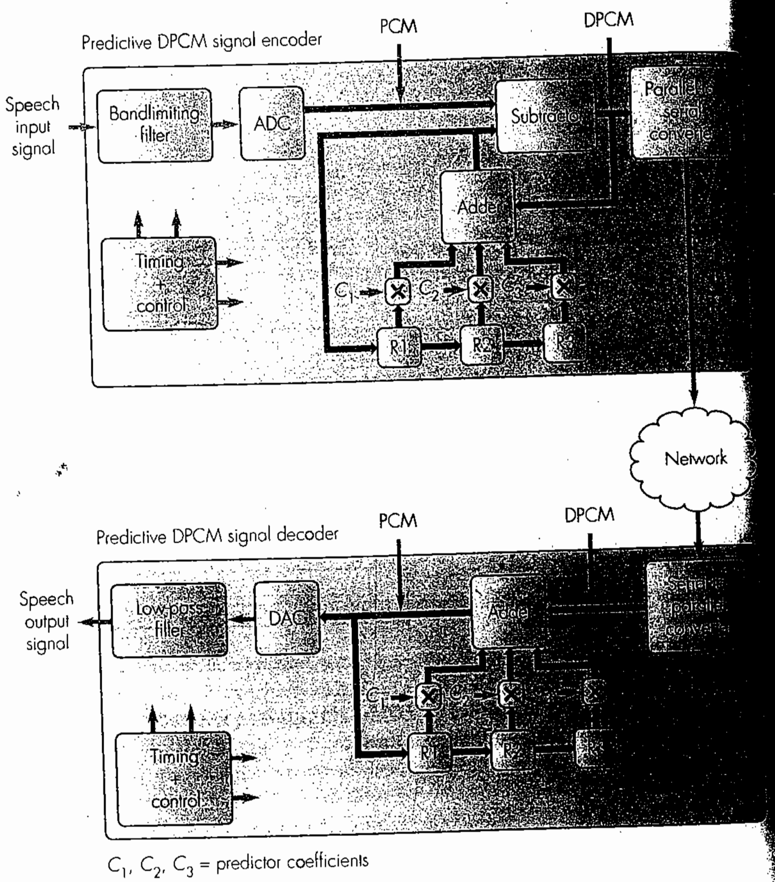


Figure 4.2 Third-order predictive DPCM signal encoder and decoder schematic.

In this example, the difference signal is computed by subtracting varying proportions of the last three predicted values from the current digitized value output by the ADC. For example, if the three predictor coefficients have the values $C_1 = 0.5$ and $C_2 = C_3 = 0.25$, then the contents of register R1 would be shifted right by 1 bit (thereby multiplying its contents by 0.5) and registers R2 and R3 by 2 bits (multiplying their contents by 0.25). The three shifted values are then added together and the resulting sum subtracted from the current digitized value output by the ADC (PCM). The current contents of register R1 are then transferred to register R2 and that of register R2 to register R3. The new predicted value is then loaded into register R1 ready for the next sample to be processed. The decoder operates in a similar way by adding the same proportions of the last three computed PCM signals to the received DPCM signal. By using this approach, a similar performance level to standard PCM is obtained by using only 6 bits for the difference signal which produces a bit rate of 32 kbps.

4.2.2 Adaptive differential PCM

Additional savings in bandwidth – or improved quality – can be obtained by varying the number of bits used for the difference signal depending on its amplitude; that is, using fewer bits to encode (and hence transmit) smaller difference values than for larger values. This is the principle of **adaptive differential PCM (ADPCM)** and an international standard for this is defined in **ITU-T Recommendation G.721**. This is based on the same principle as DPCM except an eight-order predictor is used and the number of bits used to quantize each difference value is varied. This can be either 6 bits – producing 32 kbps – to obtain a better quality output than with third-order DPCM, or 5 bits – producing 16 kbps – if lower bandwidth is more important.

A second ADPCM standard, which is a derivative of G.721, is defined in **ITU-T Recommendation G.722**. This provides better sound quality than the G.721 standard at the expense of added complexity. To achieve this, it uses an added technique known as **subband coding**. The input speech bandwidth is extended to be from 50 Hz through to 7 kHz – compared with 3.4 kHz for a standard PCM system – and hence the wider bandwidth produces a higher-fidelity speech signal. This is particularly important in conferencing applications, for example, in order to enable the members of the conference to discriminate between the different voices of the members present. The general principle of the scheme is shown in Figure 4.3.

To allow for the higher signal bandwidth, prior to sampling the audio input signal, it is first passed through two filters: one which passes only signal frequencies in the range 50 Hz through to 3.5 kHz, and the other only frequencies in the range 3.5 kHz through to 7 kHz. By doing this, the input (speech) signal is effectively divided into two separate equal-bandwidth signals, the first known as the **lower subband signal** and the second the **upper subband signal**. Each is then sampled and encoded independently using

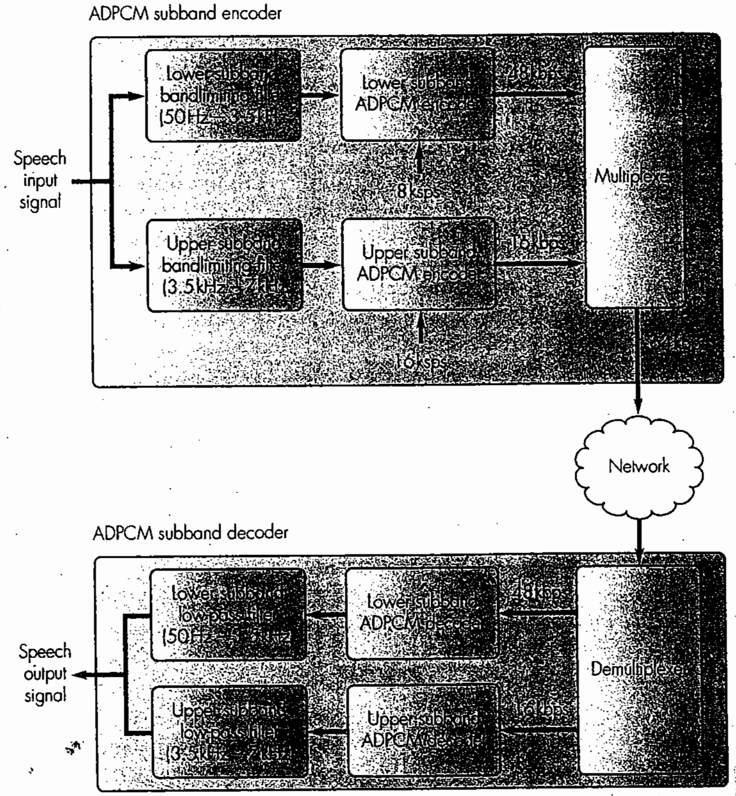

ADPCM subband encoder
Speech input signal
Lower subband bandlimiting filter (50Hz–3.5kHz)
Lower subband ADPCM encoder
48kbps
8ksps
Upper subband bandlimiting filter (3.5kHz–7kHz)
Upper subband ADPCM encoder
16kbps
16ksps
Multiplexer
Network
ADPCM subband decoder
Speech output signal
Lower subband low-pass filter (50Hz–3.5kHz)
Lower subband ADPCM decoder
48kbps
Upper subband low-pass filter (3.5kHz–7kHz)
Upper subband ADPCM decoder
16kbps
Demultiplexer

Figure 4.3 ADPCM subband encoder and decoder schematic.

ADPCM, the sampling rate of the upper subband signal being 16ksps to allo for the presence of the higher frequency components in this subband.

The use of two subbands has the advantage that different bit rates can b used for each. In general, the frequency components that are present in th lower subband signal have a higher perceptual importance than those in th higher subband. The operating bit rate can be 64, 56, or 48kbps. With a bi rate of 64kbps, for example, the lower subband is ADPCM encoded a 48kbps and the upper subband at 16kbps. The two bitstreams are then multiplexed (merged) together – to produce the transmitted (64kbps) signal – in such a way that the decoder in the receiver is able to divide them back again into two separate streams for decoding.

A third standard based on ADPCM is also available. This is defined in **ITU-T Recommendation G.726**. This also uses subband coding but with a speech bandwidth of 3.4kHz. The operating bit rate can be 40, 32, 24, or 16kbps.

4.2.3 Adaptive predictive coding

Even higher levels of compression – but at higher levels of complexity – can be obtained by also making the predictor coefficients adaptive. This is the principle of **adaptive predictive coding (APC)** and with this, the predictor coefficients continuously change.

In practice, the optimum set of predictor coefficients continuously vary since they are a function of the characteristics of the audio signal being digitized; for example, the actual frequency components that make up the signal at a particular instant of time.

To exploit this property, the input speech signal is divided into fixed time segments and, for each segment, the currently prevailing characteristics are determined. The optimum set of coefficients are then computed and these are used to predict more accurately the previous signal. This type of compression can reduce the bandwidth requirements to 8kbps while still obtaining an acceptable perceived quality.

4.2.4 Linear predictive coding

All the algorithms we have considered so far are based on sampling the time-varying speech waveform and then either sending the quantized samples directly (PCM) or sending the quantized difference signal (DPCM and its derivatives). With the advent of inexpensive digital signal processing circuits, an alternative approach has become possible which involves the source simply analyzing the audio waveform to determine a selection of the perceptual features it contains. These are then quantized and sent and the destination uses them, together with a sound synthesizer, to regenerate a sound that is perceptually comparable with the source audio signal. This is the basis of the **linear predictive coding (LPC)** technique and, although with this the generated sound – normally speech – can often sound synthetic, very high levels of compression (and hence low bit rates) can be achieved.

Clearly, the key to this approach is to identify the set of perceptual features to be used and, in terms of speech, the three features which determine the perception of a signal by the ear are its:

- pitch: this is closely related to the frequency of the signal and is important because the ear is more sensitive to frequencies in the range 2–5kHz than to frequencies that are higher or lower than these;
- period: this is the duration of the signal;
- loudness: this is determined by the amount of energy in the signal.

In addition, the origins of the sound are important. These are known as **vocal tract excitation parameters** and classified as:

- voiced sounds: these are generated through the vocal chords and examples include the sounds relating to the letters m, v, and l;

- unvoiced sounds: with these the vocal chords are open and examples include the sounds relating to the letters f and s.

Once these have been obtained from the source waveform, it is possible to use them, together with a suitable model of the vocal tract, to generate a synthesized version of the original speech signal. The basic features of an LPC encoder/decoder are shown in Figure 4.4.

The input speech waveform is first sampled and quantized at a defined rate. A block of digitized samples – known as a segment – is then analyzed to determine the various perceptual parameters of the speech that it contains. The speech signal generated by the vocal tract model in the decoder is a function of the present output of the speech synthesizer – as determined by the current set of model coefficients – plus a linear combination of the previous set of model coefficients. Hence the vocal tract model used is adaptive

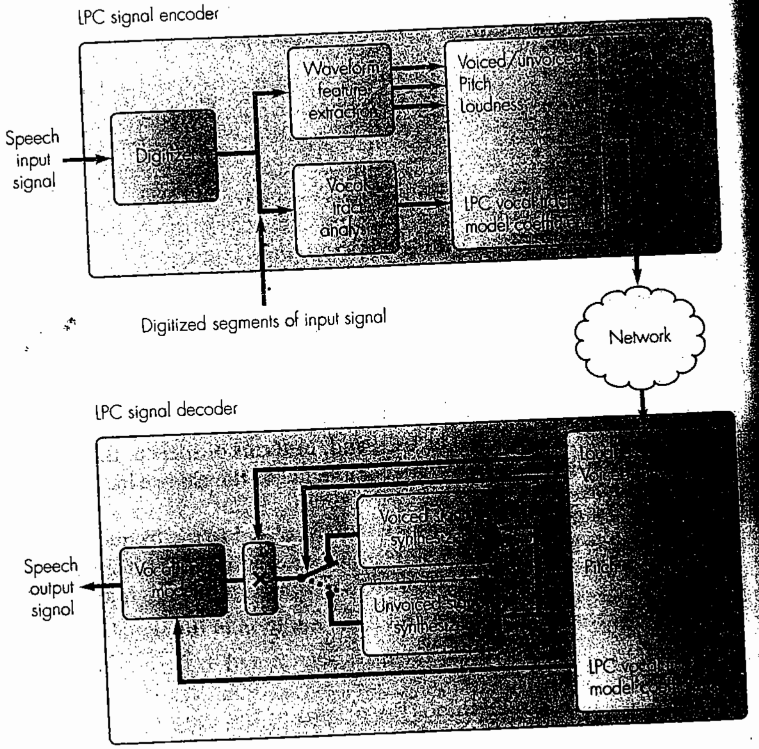


Figure 4.4 Linear predictive coding (LPC) signal encoder and decoder schematic.

and, as can be seen, the encoder determines and sends a new set of coefficients for each quantized segment.

As we can see from the above, the output of the encoder is a string of frames, one for each segment. Each frame contains fields for pitch and loudness – the period is determined by the sampling rate being used – a notification of whether the signal is voiced or unvoiced, and a new set of computed model coefficients. Some LPC encoders use up to ten sets of previous model coefficients to predict the output sound (**LPC-10**) and use bit rates as low as 2.4 kbps or even 1.2 kbps. As indicated, however, the generated sound at these rates is often very synthetic and hence LPC coders are used primarily in military applications in which bandwidth is all-important.

4.2.5 Code-excited LPC

The synthesizers used in most LPC decoders are based on a very basic model of the vocal tract. A more sophisticated version of this, known as a **code-excited linear prediction (CELP) model**, is also used and, in practice, is just one example of a family of vocal tract models known as **enhanced excitation (LPC) models**. These are also intended primarily for applications in which the amount of bandwidth available is limited but the perceived quality of the speech must be of an acceptable standard for use in various multimedia applications.

In the CELP model, instead of treating each digitized segment independently for encoding purposes, just a limited set of segments is used, each known as a **waveform template**. A precomputed set of templates are held by the encoder and decoder in what is known as a **template codebook**. Each of the individual digitized samples that make up a particular template in the codebook are differentially encoded. Each codeword that is sent selects a particular template from the codebook whose difference values best match those quantized by the encoder. In this way, there is continuity from one set of samples to another and, as a result, an improvement in sound quality is obtained.

There are now four international standards available that are based on this principle. These are **ITU-T Recommendations G.728, 729, 729(A),** and **723.1** all of which give a good perceived quality at low bit rates.

All coders of this type have a delay associated with them which is incurred while each block of digitized samples is analyzed by the encoder and the speech is reconstructed at the decoder. The combined delay value is known as the coder's **processing delay**. In addition, before the speech samples can be analyzed, it is necessary to buffer – store in memory – the block of samples. The time to accumulate the block of samples is known as the **algorithmic delay** and, in some CELP coders, this is extended to include samples from the next successive block, a technique known as **lookahead**. These delays occur in the coders, of course, and hence are in addition to the end-to-end transmission delay over the network. Nevertheless, the combined delay value of a coder is an important parameter as it often determines the suitability of the coder for a specific application. For example, in a conventional telephony application, a low-delay coder is required since a large delay can impede the flow of a conversation. In contrast, in an interactive application that involves the output of speech stored in a file, for example, a delay of several seconds before the speech starts to be output is often acceptable and hence the coder's delay is less important. Other parameters of the coder that are considered are the **complexity** of the coding algorithm and the **perceived quality** of the output speech and, in general, a compromise has to be reached between a coder's speech quality and its delay/complexity.

The delay associated with a basic PCM coder is very small as it is equal to the time interval between two successive samples of the input waveform. Hence at the basic PCM sampling rate of 8 ksps the delay is equal to 0.125 ms. This same delay also applies, of course, to ADPCM coders. In contrast, the four CELP-based standards have delay values in excess of these as multiple samples are involved. These are summarized in Table 4.1 which also includes the bit rate(s) associated with each standard and the principal application for which each has been developed. The use of the extension .1 with G.723.1 is used to discriminate this standard from the earlier G.723 standard which has now been integrated with G.721 into the G.726 standard.

.1 Summary of CELP-based standards.

rd	Bit rate	Total coder delay	Example application domain
	16 kbps	0.625 ms	Low bit rate telephony
	8 kbps	25 ms	Telephony in cellular (radio) networks
A	8 kbps	25 ms	Digital simultaneous voice and data (DSVD)
	5.3/6.3 kbps	67.5 ms	Video and Internet telephony

4.2.6 Perceptual coding

Both LPC and CELP are used primarily for telephony applications and hence the compression of a speech signal. Perceptual encoders, however, have been designed for the compression of general audio such as that associated with a digital television broadcast. They also use a model but, in this case, it is known as a **psychoacoustic model** since its role is to exploit a number of the limitations of the human ear.

Using this approach, sampled segments of the source audio waveform are analyzed – as with CELP-based coders – but only those features that are perceptible to the ear are transmitted. For example, although the human ear is sensitive to signals in the range 15 Hz through to 20 kHz, the level of sensitivity to each signal is non-linear; that is, the ear is more sensitive to some signals

than others. Also, when multiple signals are present – as is the case with general audio – a strong signal may reduce the level of sensitivity of the ear to other signals which are near to it in frequency, an effect known as **frequency masking**. In addition, when the ear hears a loud sound, it takes a short but finite time before it can hear a quieter sound, an effect known as **temporal masking**. A psychoacoustic model is used to identify those signals that are influenced by both these effects. These are then eliminated from the transmitted signals and, in so doing, this reduces the amount of information to be transmitted.

Sensitivity of the ear

We defined the dynamic range of a signal in Section 2.2.2. It is the ratio of the maximum amplitude of the signal to the minimum amplitude and is measured in decibels (dB). In the case of the ear, its dynamic range is the ratio of the loudest sound it can hear to the quietest sound and is in the region of 96 dB. As we have just seen, however, the sensitivity of the ear varies with the frequency of the signal and, assuming just a single-frequency signal is present at any one time, the perception threshold of the ear – that is, its minimum level of sensitivity – as a function of frequency is shown in Figure 4.5(a).

As we can see, the ear is most sensitive to signals in the range 2–5 kHz and hence signals within this band are the quietest the ear is sensitive to. The vertical axis, therefore, indicates the amplitude level of all the other signal frequencies relative to this level – measured in dB – that are required for them to be heard. Hence in the figure, although the two signals A and B have the same relative amplitude, signal A would be heard – that is, it is above the hearing threshold – while signal B would not.

Frequency masking

When an audio sound consists of multiple frequency signals is present, the sensitivity of the ear changes and varies with the relative amplitude of the signals. For example, the curve shown in Figure 4.5(b) shows how the sensitivity of the ear changes in the vicinity of a loud signal. In this example, signal B is larger in amplitude than signal A and, as we can see, this causes the basic sensitivity curve of the ear to be distorted in the region of signal B. As a result, signal A will no longer be heard even though on its own, it is above the hearing threshold of the ear for a signal of that frequency. This is the origin of the term frequency masking and, in practice, the masking effect also varies with frequency as we show in Figure 4.6.

The various curves show the masking effect of a selection of different frequency signals – 1, 4, and 8 kHz – and, as we can see, the width of the masking curves – that is, the range of frequencies that are affected – increase with increasing frequency. The width of each curve at a particular signal level is known as the **critical bandwidth** for that frequency and experiments have shown that, for frequencies less than 500 Hz, the critical bandwidth remains constant at about 100 Hz. For frequencies greater than 500 Hz, however, the critical bandwidth increases (approximately) linearly in multiples of 100 Hz. For example, for a signal of 1 kHz (2 × 500 Hz), the critical bandwidth is

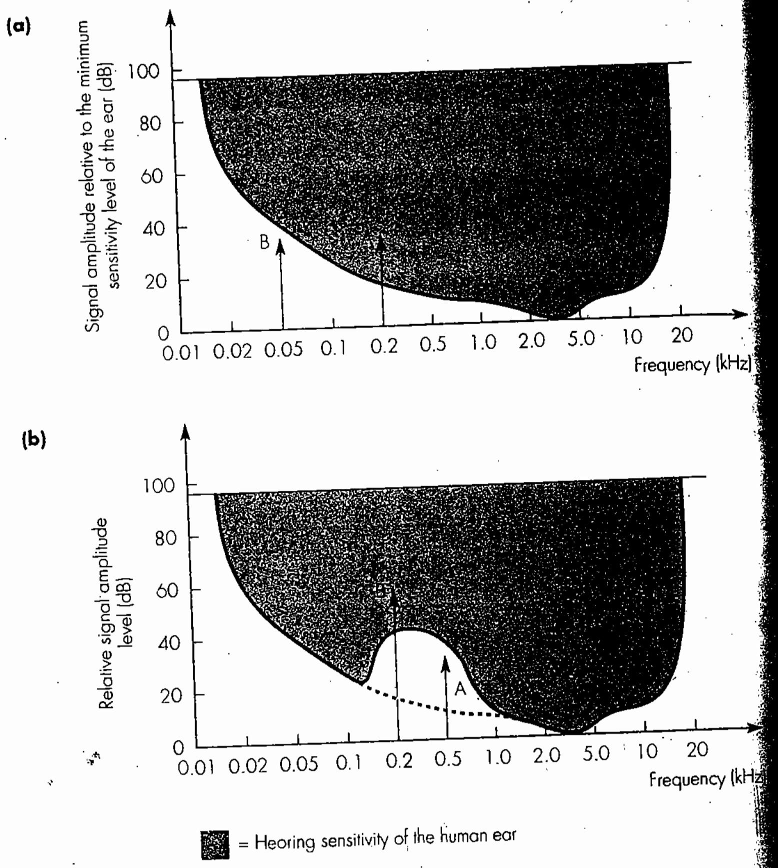


Figure 4.5 Perceptual properties of the human ear: (a) sensitivity as function of frequency; (b) frequency masking.

about 200 (2 × 100) Hz while at 5 kHz (10 × 500 Hz) it is about 1000 (10 100) Hz. Hence if the magnitude of the frequency components that make u an audio sound can be determined, it becomes possible to determine tho frequencies that will be masked (and hence inaudible) and do not therefo need to be transmitted.

Temporal masking

As indicated earlier, after the ear hears a loud sound, it takes a further shor time before it can hear a quieter sound. This is known as temporal masking and the general effect is shown in Figure 4.7. As we can see, after the loud

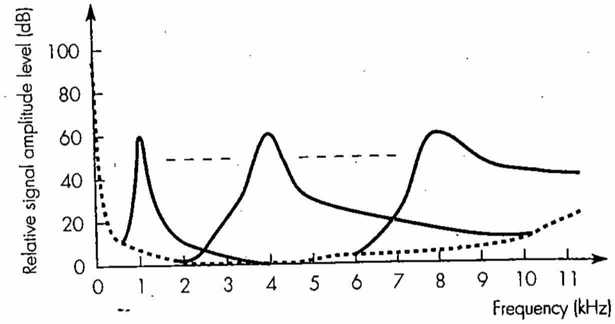


Figure 4.6 Variation with frequency of effect of frequency masking.

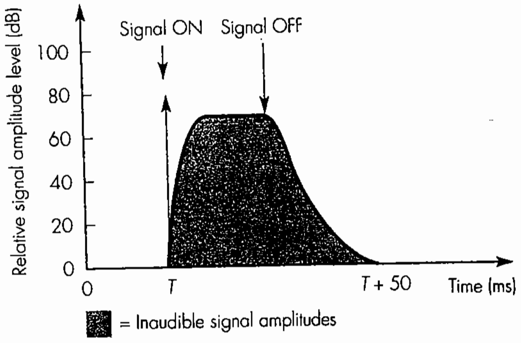


Figure 4.7 Temporal masking caused by a loud signal.

sound ceases it takes a short period of time (in the order of tens of milliseconds) for the signal amplitude to decay. During this time, signals whose amplitudes are less than the decay envelope will not be heard and hence need not be transmitted. Clearly, however, in order to exploit this phenomenon, it is necessary to process the input audio waveform over a time period that is comparable with that associated with temporal masking.

4.2.7 MPEG audio coders

In practice, perceptual coding is used in a range of different audio compression applications. For example, as we shall describe later in Section 4.3.4, the **Motion Pictures Expert Group (MPEG)** was formed by the ISO to formulate a set of standards relating to a range of multimedia applications that involve the use of video with sound. The coders associated with the audio compression part of these standards are known as **MPEG audio coders** and a number of these use perceptual coding.

All the signal processing operations associated with a perceptual coder are carried out digitally and a schematic diagram of a basic encoder and decoder is shown in Figure 4.8(a).

The time-varying audio input signal is first sampled and quantized using PCM, the sampling rate and number of bits per sample being determined by the specific application. The bandwidth that is available for transmission is divided into a number of **frequency subbands** using a bank of **analysis filters** which, because of their role, are also known as **critical-band filters**. Each frequency subband is of equal width and, essentially, the bank of filters maps each set of 32 (time-related) PCM samples into an equivalent set of 32 frequency samples, one per subband. Hence each is known as a **subband sample** and indicates the magnitude of each of the 32 frequency components that are present in a segment of the audio input signal of a time duration equal to 32 PCM samples. For example, assuming 32 subbands and a sampling rate of 32 ksps – that is, a maximum signal frequency of 16 kHz – each subband has a bandwidth of 500 Hz.

In a basic encoder, the time duration of each sampled segment of the audio input signal is equal to the time to accumulate 12 successive sets of 32 PCM – and hence subband – samples; that is, a time duration equal to 384 (12 x 32) PCM samples.

In addition to filtering the input samples into separate frequency subbands, the analysis filter bank also determines the maximum amplitude of the 12 subband samples in each subband. Each is known as the **scaling factor** for the subband and these are passed both to the psychoacoustic model and together with the set of frequency samples in each subband, to the corresponding quantizer block.

The processing associated with both frequency and temporal masking i carried out by the psychoacoustic model which is performed concurrentl with the filtering and analysis operations. The 12 sets of 32 PCM samples a first transformed into an equivalent set of frequency components using mathematical technique known as the **discrete Fourier transform (DFT)**. Then, using the known hearing thresholds and masking properties of eac subband, the model determines the various masking effects of this set of signals. The output of the model is a set of what are known as **signal-to-mas ratios (SMRs)** and indicate those frequency components whose amplitude i below the related audible threshold. In addition, the set of scaling factors are used to determine the quantization accuracy – and hence bit allocations – to be used for each of the audible components. This is done so that those frequency components that are in regions of highest sensitivity can be quantized with more accuracy (bits) – and hence less quantization noise – than those in regions where the ear is less sensitive. In a basic encoder, all the frequency components in a sampled segment are encoded and these are carried in a frame the format of which is shown in Figure 4.8(b).

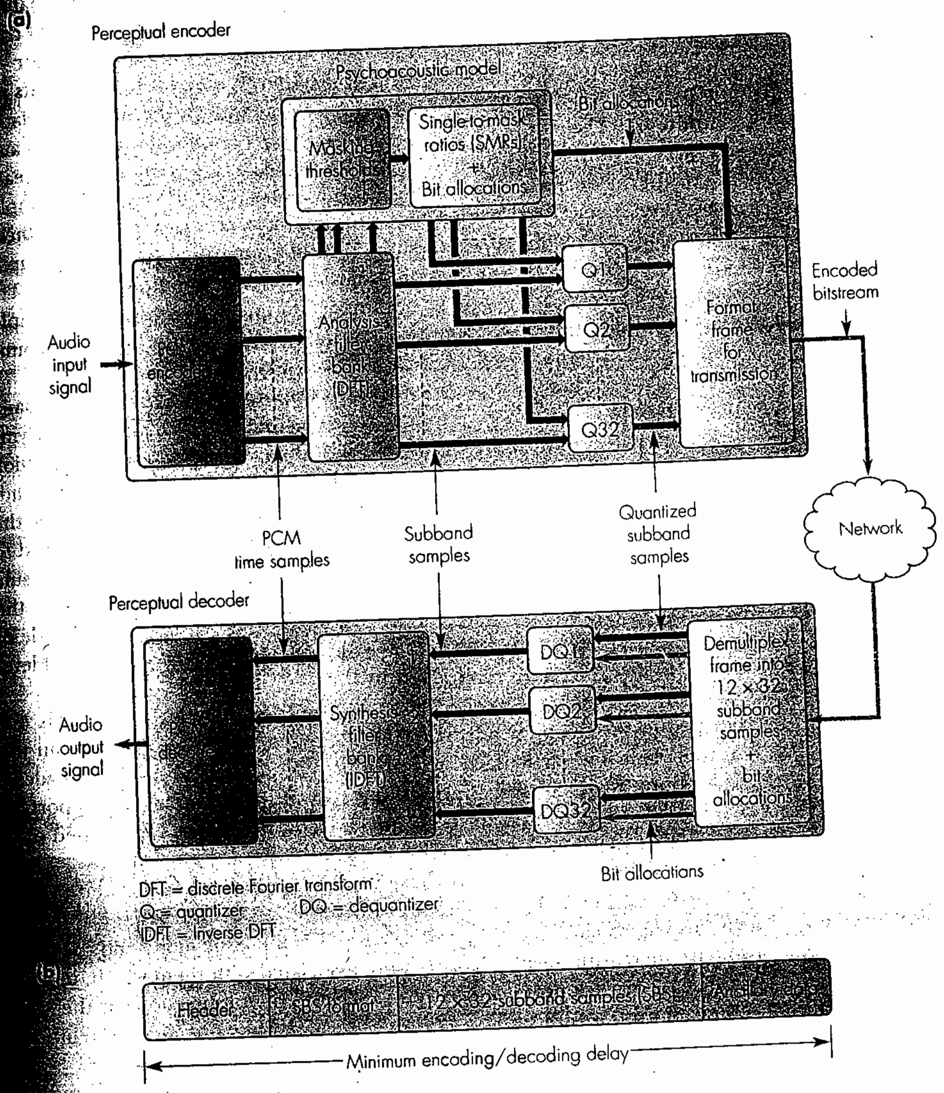

(a)
Perceptual encoder
Psychoacoustic model
Masking thresholds
Single-to-mask ratios (SMRs) + Bit allocations
Bit allocations
Analysis filter bank (DFT)
encoder
Q1
Q2
Q32
Format frame for transmission
Encoded bitstream
Audio input signal
PCM time samples
Subband samples
Quantized subband samples
Network
Perceptual decoder
Synthesis filter bank (IDFT)
DQ1
DQ2
DQ32
Demultiplex frame into 12 × 32 subband samples + bit allocations
Audio output signal
Bit allocations
DFT = discrete Fourier transform
Q = quantizer
DQ = dequantizer
IDFT = Inverse DFT
(b)
Header
SBS format
12 × 32 subband samples (SBS)
Minimum encoding/decoding delay

Figure 4.8 MPEG perceptual coder schematic: (a) encoder/decoder implementation schematic (b) example frame format.

The *header* contains information such as the sampling frequency that has been used. The quantization is performed in two stages using a form of companding. The peak amplitude level in each subband – the scaling factor – is first quantized using 6 bits – giving 1 of 64 levels – and a further 4 bits are then used to quantize the 12 frequency components in the subband relative to this level. Collectively this is known as the ***subband sample (SBS) format*** and, in this way, all the information necessary for decoding is carried within each frame. In the decoder, after the magnitude of each set of 32 subband samples have been determined by the dequantizers, these are passed to the **synthesis filter** bank. The latter then produces the corresponding set of PCM samples which are decoded to produce the time-varying analog output segment. The *ancillary data* field at the end of a frame is optional and is used, for example, to carry additional coded samples associated with, say, the surround-sound that is present with some digital video broadcasts.

The use in the encoder of different scaling factors for each subband means that the frequency components in the different subbands have varying levels of quantization noise associated with them. As we described in Section 2.2.2, this means that the frequency components in the different subbands have varying signal-to-noise ratios. The bank of synthesis filters in the decoder, however, limits the level of quantization noise in each subband to the same band of frequencies as the set of frequency components in that subband. As a result, the effect of quantization noise is reduced since the signal-to-noise ratio in each subband is increased by the larger amplitude of the signal frequency components in each subband masking the reduced level of quantization noise that is present.

As we can deduce from the figure, the psychoacoustic model is not required in the decoder and, as a consequence, it is less complex than the encoder. This is a particularly desirable feature in audio and video broadcast applications, for example, in which the cost of the decoder is an important factor. Also, it means that different psychoacoustic models can be used or, if bandwidth is plentiful, none at all.

An international standard based on this approach is defined in **ISO Recommendation 11172-3**. There are three levels of processing associated with this known as layers 1, 2, and 3. Layer 1 is the basic mode and the other two have increasing levels of processing associated with them which, in turn, produce a corresponding increase in the level of compression for the same perceived audio quality. For example, layer 1 does not include temporal masking but this is present in layers 2 and 3.

As we have already indicated, in terms of applications, MPEG audio is used primarily for the compression of general audio and, in particular, for the audio associated with various digital video applications. The performance of the three layers and examples of their corresponding application domains are summarized in Table 4.2. As we can see, the encoders associated with each of the three layers obtain increasing levels of compression and perceptual quality. The encoder and decoder delay figures are determined by the PCM sampling rate used and the corresponding frame size. For example, with layer 1, the sampling rates used are:

- 32 ksps for use with broadcast communications equipment,
- 44.1 ksps for use with CD-quality audio equipment,
- 48 ksps for use with professional sound equipment.

These produce corresponding frame durations of 12, 8.7, and 8 milliseconds with 384 samples per frame. The actual input-to-output delay, however, can be as much as two to three times these values owing to additional processing delays in the encoder and decoder. Layer 2 is identical to a standard known as **MUSICAM** and indeed was based on this. The format of each layer 2 frame is similar to that of layer 1 except that each frame contains three sets of 384 samples and hence is of a time duration equal to 1152 samples.

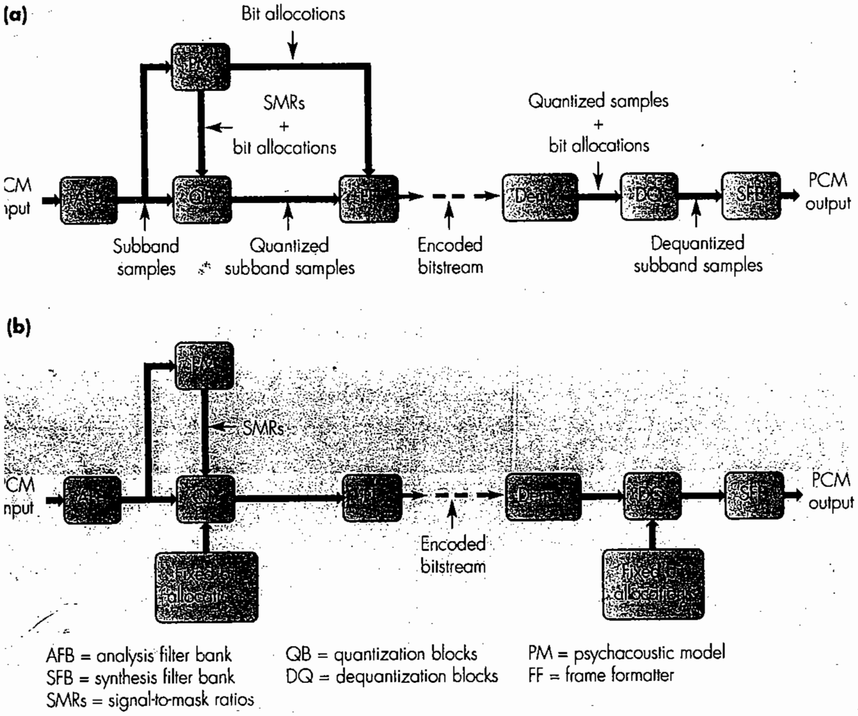
The bit rate figures shown in Table 4.2 are all for a single audio channel. In practice, four alternative forms of audio have been identified for multimedia applications: monophonic, dual monophonic, two-channel (also known as disjoint) stereo, and single-channel joint stereo. The latter is the digitized version of the composite stereo sound signal and hence exploits the redundancy that is present between the two channels. So the bandwidth required for audio may be the figures shown in the table – for monophonic and joint stereo – or double the values shown – for dual monophonic and two-channel stereo. Since the three layers require increasing levels of complexity (and hence cost) to achieve a particular perceived quality, the choice of layer and bit rate is often a compromise between the desired perceived quality and the available bit rate.

Table 4.2 Summary of MPEG layer 1, 2 and 3 perceptual encoders

Layer	Application	Compressed bit rate	Quality	Example input-to-output delay
[illegible]	Digital audio cassette	32 – 448 kbps	Hi-fi quality at 192 kbps per channel	20ms
[illegible]	Digital audio and digital video broadcasting	32 – 192 kbps	Near CD-quality at 128 kbps per channel	40ms
[illegible]	CD-quality audio over low bit rate channels	64 kbps	CD-quality at 64 kbps per channel	60ms

4.2.8 Dolby audio coders

The psychoacoustic models associated with the various MPEG coders control the quantization accuracy of each subband sample by computing and allocating the number of bits to be used to quantize each sample. Since the quantization accuracy that is used for each sample in a subband may vary from one set of subband samples to the next, the bit allocation information that is used to quantize the samples in each subband is sent with the actual quantized samples. This information is then used by the decoder to dequantize the set of subband samples in the frame. This mode of operation of a perceptual coder is known, therefore, as the **forward adaptive bit allocation mode** and, for comparison purposes, a simplified schematic diagram showing this operational mode is given in Figure 4.9(a). As we indicated at the end of the last section, it has the advantage that the psychoacoustic model is required only in the encoder. It has the disadvantage, however, that a significant portion of each encoded frame contains bit allocation information which, in turn, leads to a relatively inefficient use of the available bit rate.



4.9 Perceptual coder schematics: (a) forward adaptive bit allocation (MPEG); ed bit allocation (Dolby AC-1).

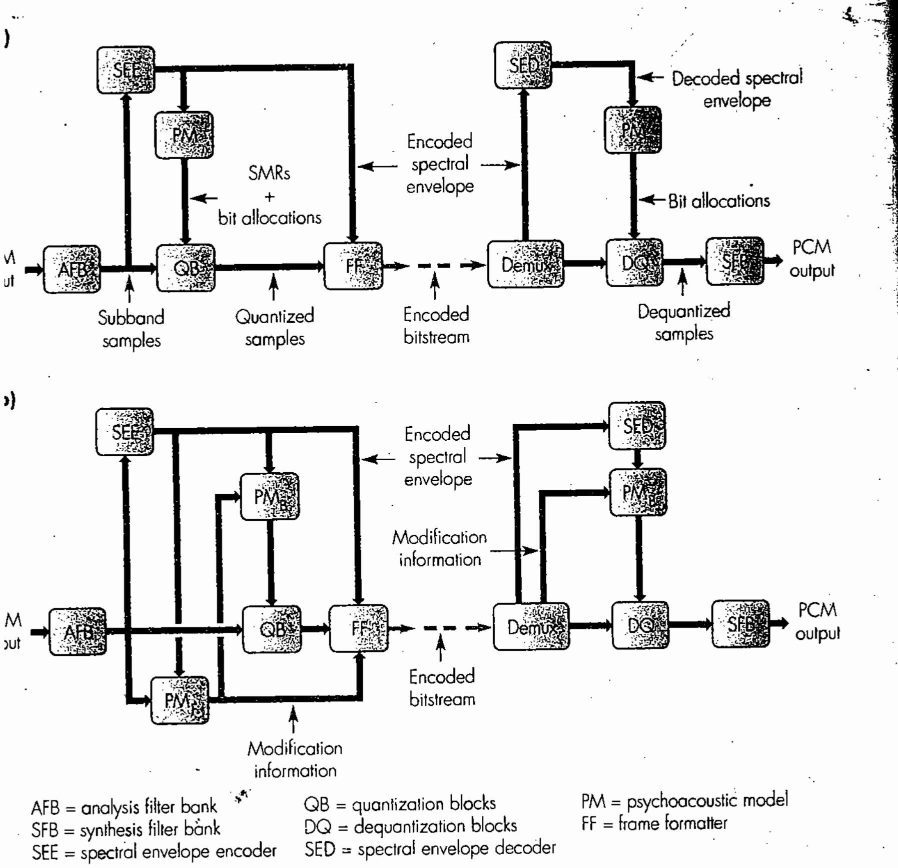
A variation of this approach is to use a fixed bit allocation strategy for each subband which is then used by both the encoder and decoder. The principle of operation of this mode is shown in Figure 4.9(b). Typically, the bit allocations that are selected for each subband are determined by the known sensitivity characteristics of the ear and the use of fixed allocations means that this information need not be sent in the frame. This approach is used in a standard known as **Dolby AC-1**, the acronym "AC" meaning **acoustic coder**. It was designed for use in satellites to relay FM radio programs and the sound associated with television programs. It uses a low-complexity psychoacoustic model with 40 subbands at a sampling rate of 32 ksps and proportionately more at 44.1 and 48 ksps. A typical compressed bit rate is 512 kbps for two-channel stereo.

A second variation, which allows the bit allocations per subband to be adaptive while at the same time minimizing the overheads in the encoder bitstream, is for the decoder also to contain a copy of the psychoacoustic model. This is then used by the decoder to compute the same – or very similar – bit allocations that the psychoacoustic model in the encoder has used to quantize each set of subband samples. Clearly, however, in order for the psychoacoustic model in the decoder to carry out its own computation of the bit allocations, it is necessary for it to have a copy of the subband samples. Hence with this operational mode, instead of each frame containing bit allocation information – in addition to the set of quantized samples – it contains the encoded frequency coefficients that are present in the sampled waveform segment. This is known as the **encoded spectral envelope** and this mode of operation, the **backward adaptive bit allocation mode**. Figure 4.10(a) illustrates the principle of operation of both the encoder and decoder.

This approach is used in the **Dolby AC-2** standard which is utilized in many applications including the compression associated with the audio of a number of PC sound cards. Typically, these produce audio of hi-fi quality at a bit rate of 256 kbps. For broadcast applications, however, it has the disadvantage that, since the same psychoacoustic model is required in the decoder, the model in the encoder cannot be modified without changing all decoders.

To meet this requirement, a third variation has been developed that uses both backward and forward bit allocation principles. This is known as the **hybrid backward/forward adaptive bit allocation mode** and is illustrated in Figure 4.10(b).

As we can deduce from part (a) of the figure, with the backward bit allocation method on its own, since the psychoacoustic model uses the encoded spectral envelope, the quantization accuracy of the subband samples is affected by the quantization noise introduced by the spectral encoder. Hence in the hybrid scheme, although a backward adaptive bit allocation scheme is used as in AC-2 – using PM_B – an additional psychoacoustic model – PM_F – is used to compute the difference between the bit allocations computed by PM_B and those that are computed by PM_F using the forward-adaptive bit allocation scheme. This information is then used by PM_B to improve the quantization


)
SEE
PM
SMRs
+
bit allocations
Encoded
spectral
envelope
SED
Decoded spectral
envelope
PM
Bit allocations
AFB
QB
FF
Demux
DQ
SFB
PCM
output
Subband
samples
Quantized
samples
Encoded
bitstream
Dequantized
samples
›)
SEE
PM
Encoded
spectral
envelope
SED
PM
Modification
information
AFB
QB
FF
Demux
DQ
SFB
PCM
output
PM
Encoded
bitstream
Modification
information
AFB = analysis filter bank
SFB = synthesis filter bank
SEE = spectral envelope encoder
QB = quantization blocks
DQ = dequantization blocks
SED = spectral envelope decoder
PM = psychoacoustic model
FF = frame formatter

4.10 Perceptual coder schematic: (a) backward adaptive bit allocation (Dolby AC-2); rid backward/forward adaptive bit allocation (Dolby AC-s).

accuracy of the set of subband samples. The modification information is also sent in the encoded frame and is used by the PMB in the decoder to improve the dequantization accuracy. In addition, should it be required to modify the operational parameters of the PMB in the encoder and decoder(s), then this information can be sent also with the computed difference information. As we can see from the figure, the PMF must compute two sets of quantization information for each set of subband samples and hence is relatively complex. However, since this is not required in the decoder, this is not an issue.

The hybrid approach is used in the **Dolby AC-3** standard which has been defined for use in a similar range of applications as the MPEG audio standards including the audio associated with **advanced television** (ATV). This is the HDTV standard in North America and, in this application, the acoustic quality of both the MPEG and Dolby audio coders were found to be comparable. The sampling rate can be 32, 44.1, or 48 ksps depending on the bandwidth of the source audio signal. Each encoded block contains 512 subband samples. However, in order to obtain continuity from one block to the next, the last 256 subband samples in the previous block are repeated to become the first 256 samples in the next block and hence each block contains only 256 new samples. Assuming a PCM sampling rate of 32 ksps, although each block of samples is of 8 ms duration – 256/32 – the duration of each encoder block is 16 ms. The audio signal bandwidth at this sampling rate is 15 kHz and hence each subband has a bandwidth of 62.5 Hz, that is, 15 k/256. The (stereo) bit rate is, typically, 192 kbps.

4.3 Video compression

As we described in Chapter 1, video (with sound) features in a number of multimedia applications:

- interpersonal: video telephony and videoconferencing;
- interactive: access to stored video in various forms;
- entertainment: digital television and movie/video-on-demand.

However, as we described in Section 2.6.2, the quality of the video used in these applications varies and is determined by the digitization format and frame refresh rate used. As you may recall, the digitization format defines the sampling rate that is used for the luminance, Y, and two chrominance, C_b and C_r, signals and their relative position in each frame. We derived a number of (worst-case) bit rates that are generated with different digitization formats and these ranged from in the order of 10 Mbps for the sub-quarter common intermediate format (SQCIF), as used for video telephony, to 162 Mbps for the 4:2:0 format as used for digital television broadcasts.

In practice, therefore, as we described in Chapter 1, the bit rate requirements resulting from all the digitization formats defined in Section 2.6.2 are substantially larger than the bit rates of the transmission channels that are available with the networks associated with these applications. It is for this reason that compression is used in all applications that involve video. However, there is not just a single standard associated with video but rather a range of standards, each targeted at a particular application domain. The majority are all now international standards and we shall describe a number of these in this section. Prior to doing this, however, we shall first describe the basic principles on which the standards are based.

4.3.1 Video compression principles

In the context of compression, since video is simply a sequence of digitized pictures, video is also referred to as **moving pictures** and the terms "frame" and "picture" are used interchangeably. In general, we shall use the term frame except where a particular standard uses the term picture.

In principle, one approach to compressing a video source is to apply the JPEG algorithm described earlier in Section 3.4.5 to each frame independently. This approach is known as **moving JPEG** or **MJPEG**. As we concluded at the end of that section, however, typical compression ratios obtainable with JPEG are between 10:1 and 20:1, neither of which is large enough on its own to produce the compression ratios needed.

In practice, in addition to the spatial redundancy present in each frame, considerable redundancy is often present between a set of frames since, in general, only a small portion of each frame is involved with any motion that is taking place. Examples include the movement of a person's lips or eyes in a video telephony application and a person or vehicle moving across the screen in a movie. In the case of the latter, since a typical scene in a movie has a minimum duration of about 3 seconds, assuming a frame refresh rate of 60 frames per second, each scene is composed of a minimum of 180 frames. Hence by sending only information relating to those segments of each frame that have movement associated with them, considerable additional savings in bandwidth can be made by exploiting the temporal differences that exist between many of the frames.

The technique that is used to exploit the high correlation between successive frames is to predict the content of many of the frames. As we shall describe, this is based on a combination of a preceding – and in some instances a succeeding – frame. Instead of sending the source video as a set of individually compressed frames, just a selection is sent in this form and, for the remaining frames, only the differences between the actual frame contents and the predicted frame contents are sent. The accuracy of the prediction operation is determined by how well any movement between successive frames is estimated. This operation is known as **motion estimation** and, since the estimation process is not exact, additional information must also be sent to indicate any small differences between the predicted and actual positions of the moving segments involved. The latter is known as **motion compensation** and we shall discuss each issue separately.

Frame types

As we can see from the above, there are two basic types of compressed frame: those that are encoded independently and those that are predicted. The first are known as **intracoded frames** or **I-frames**. In practice, there are two types of predicted frames: **predictive** or **P-frames** and **bidirectional** or **B-frames** and because of the way they are derived, the latter are also known as **intercoded** or **interpolation frames**. A typical sequence of frames involving just I- and P-frames is shown in Figure 4.11(a) and a sequence involving all three frame types is shown in part (b) of the figure.

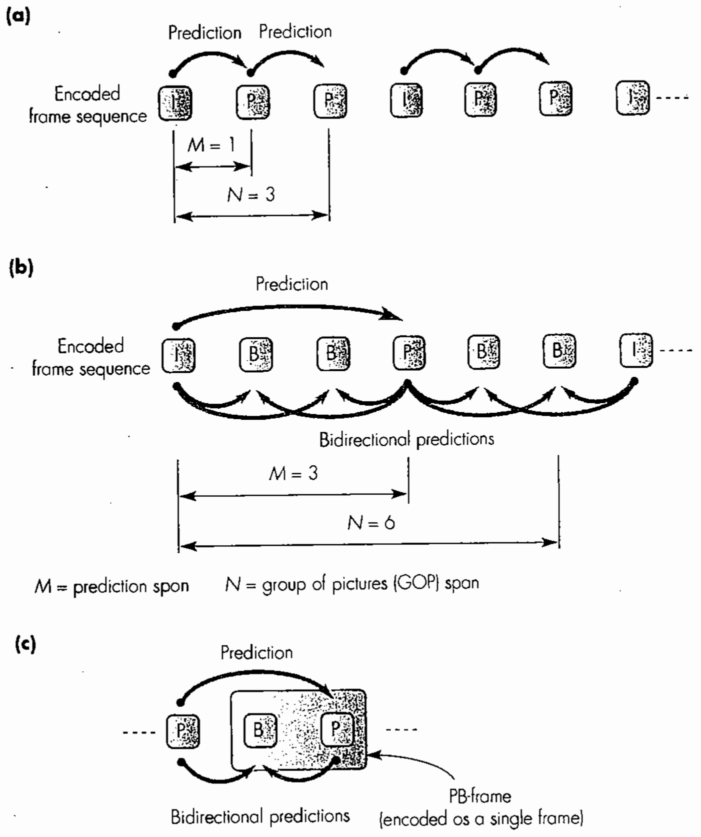


Figure 4.11 Example frame sequences with: (a) I- and P-frames only; (b) I-, P- and B-frames; (c) PB-frames.

I-frames are encoded without reference to any other frames. Each frame is treated as a separate (digitized) picture and the Y, C_b and C_r matrices are encoded independently using the JPEG algorithm (DCT, quantization, entropy encoding – described earlier in Section 3.4.5) except that the quantization threshold values that are used are the same for all DCT coefficients. Hence the level of compression obtained with I-frames is relatively small. In principle, therefore, it would appear to be best if these were limited to, say, the first frame relating to a new scene in a movie. In practice, however, the compression algorithm is independent of the contents of frames and hence has no knowledge of the start and end of scenes. Also, I-frames must be present in the output stream at regular intervals in order to allow for the possibility of the contents of an encoded I-frame being corrupted durin transmission. Clearly, if an I-frame was corrupted then, in the case of a movie since the predicted frames are based on the contents of an I-frame, a com plete scene would be lost which, of course, would be totally unacceptable Normally, therefore, I-frames are inserted into the output stream relativel frequently. The number of frames/pictures between successive I-frames i known as a **group of pictures** or **GOP**. It is given the symbol *N* and typica values for *N* are from 3 through to 12.

As we can deduce from Figure 4.11(a), the encoding of a P-frame is relative to the contents of either a preceding I-frame or a preceding P-frame. A indicated, P-frames are encoded using a combination of motion estimatio and motion compensation and hence significantly higher levels of compression can be obtained with them. In practice, however, the number of P-frame between each successive pair of I-frames is limited since any errors present i the first P-frame will be propagated to the next. The number of frame between a P-frame and the immediately preceding I- or P-frame is called th **prediction span**. It is given the symbol *M* and typical values range from 1, a in Figure 4.11(a), through to 3, as in Figure 4.11(b).

As we shall describe shortly, motion estimation involves comparing small segments of two consecutive frames for differences and, should a difference be detected, a search is carried out to determine to which neighbouring segment the original segment has moved. In order to minimize the time for each search, the search region is limited to just a few neighbouring segments. In applications such as video telephony, the amount of movement between consecutive frames is relatively small and hence this approach works well. In those applications that may involve very fast moving objects, however, it is possible for a segment to have moved outside of the search region. To allow for this possibility, in applications such as movies, in addition to P-frames, second type of prediction frame is used. These are the B-frames and, as we can see in Figure 4.11(b), their contents are predicted using search regions in both past and future frames. In addition to allowing for occasional fas moving objects, this also provides better motion estimation when, for example, an object moves in front of or behind another object.

As we can deduce from the two example frame sequences shown i Figure 4.11, for P-frames their contents are encoded by considering the co tents of the current (uncoded) frame relative to the contents of th immediately preceding (uncoded) frame. In the case of B-frames, howeve three (uncoded) frame contents are involved: the immediately preceding or P-frame, the current frame being encoded, and the immediately succeeding I- or P-frame. This results in an increase in the encoding (and decoding delay which is equal to the time to wait for the next I- or P-frame in th sequence. In practice, however, B-frames provide the highest level of compression and, because they are not involved in the coding of other frames they do not propagate errors.

To perform the decoding operation, the received (compressed) information relating to I-frames can be decoded immediately it is received in order to

recreate the original frame. With P-frames, the received information is first decoded and the resulting information is then used, together with the decoded contents of the preceding I- or P-frame, to derive the decoded frame contents. In the case of B-frames, the received information is first decoded and the resulting information is then used, together with both the immediately preceding I- or P-frame contents and the immediately succeeding P- or I-frame contents, to derive the decoded frame contents. Hence in order to minimize the time required to decode each B-frame, the order of encoding (and transmission) of the (encoded) frames is changed so that both the preceding and succeeding I- or P-frames are available when the B-frame is received. For example, if the uncoded frame sequence is:

IBBPBBPBBI...

then the reordered sequence would be:

IPBBPBBIBBPBB.

Hence, with B-frames, the decoded contents of both the immediately preceding I- or P-frame and the immediately succeeding P- or I-frame are available when the B-frame is received.

A fourth type of frame known as a PB-frame has also been defined. This does not refer to a new frame type as such but rather the way two neighbouring P- and B-frames are encoded as if they were a single frame. The general principle is as shown in Figure 4.11(c) and, as we shall expand upon in Section 4.3.3, this is done in order to increase the frame rate without significantly increasing the resulting bit rate required.

Finally, although only used in a specific type of application, a fifth type of frame known as a **D-frame** has been defined for use in movie/video-on-demand applications. As we described in Section 1.4.3, with this type of application, a user (at home) can select and watch a particular movie/video which is stored in a remote server connected to a network. The selection operation is performed by means of a set-top box and, as with a VCR, the user may wish to rewind or fast-forward through the movie. Clearly, this requires the compressed video to be decompressed at much higher speeds. Hence to support this function, the encoded video stream also contains D-frames which are inserted at regular intervals throughout the stream. These are highly compressed frames and are ignored during the decoding of P- and B-frames. As you may recall from our earlier discussion of the DCT compression algorithm in Section 3.4.5, the DC coefficient associated with each 8 × 8 block of pixels – both for the luminance and the two chrominance signals – is the mean of all the values in the related block. Hence by using only the encoded DC coefficients of each block of pixels in the periodically inserted D-frames, a low-resolution sequence of frames is provided each of which can be decoded at the higher speeds that are expected with the rewind and fast-forward operations.

Motion estimation and compensation

As we showed earlier in Figure 4.11, the encoded contents of both P- and B-frames are predicted by estimating any motion that has taken place between the frame being encoded and the preceding I- or P-frame and, in the case of B-frames, the succeeding P- or I-frame. The various steps that are involved in encoding each P-frame are shown in Figure 4.12.

As we show in Figure 4.12(a), the digitized contents of the *Y* matrix associated with each frame are first divided into a two-dimensional matrix of

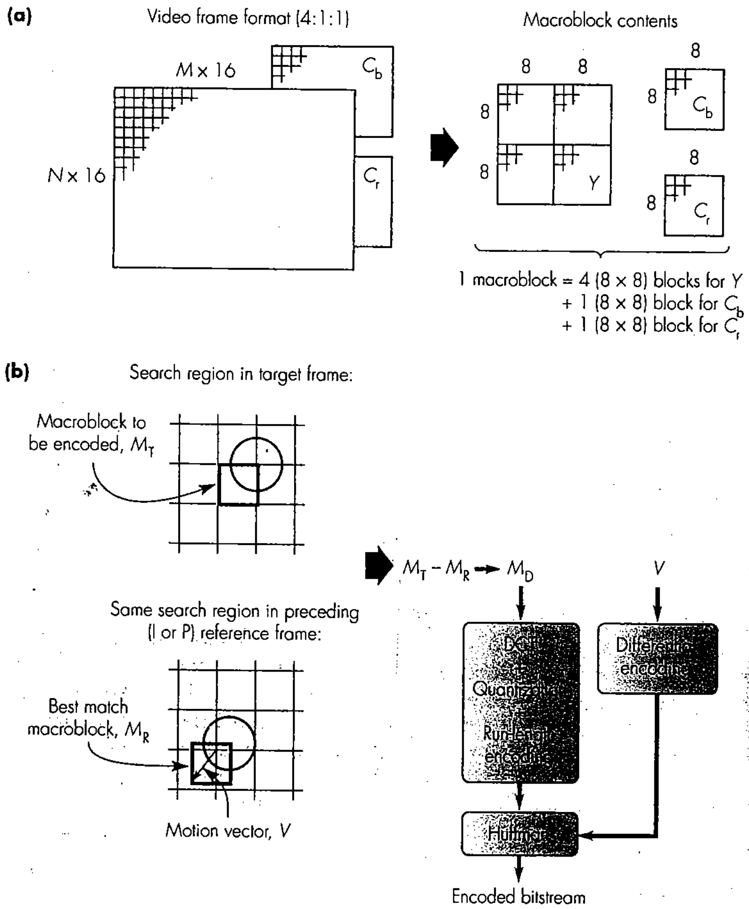

(a)
Video frame format (4:1:1)
M × 16
N × 16
C_b
C_r
Macroblock contents
8
8
8
8
8
8
8
8
C_b
C_r
Y
1 macroblock = 4 (8 × 8) blocks for Y
+ 1 (8 × 8) block for C_b
+ 1 (8 × 8) block for C_r
(b)
Search region in target frame:
Macroblock to be encoded, M_T
Same search region in preceding (I or P) reference frame:
Best match macroblock, M_R
Motion vector, V
M_T – M_R → M_D
V
DCT
+
Quantization
+
Run-length encoding
Differential encoding
Huffman
Encoded bitstream

Figure 4.12 P-frame encoding: (a) macroblock structure; (b) encoding procedure.

16 ×16 pixels known as a **macroblock**. In the example, the 4:1:1 digitization format is assumed and hence the related C_b and C_r matrices in the macroblock are both 8 × 8 pixels. For identification purposes, each macroblock has an **address** associated with it and, since the block size used for the DCT operation is also 8 × 8 pixels, a macroblock comprises four DCT blocks for luminance and one each for the two chrominance signals.

To encode a P-frame, the contents of each macroblock in the frame – known as the **target frame** – are compared on a pixel-by-pixel basis with the contents of the corresponding macroblock in the preceding -I or P-frame. The latter is known as the **reference frame.** If a close match is found, then only the address of the macroblock is encoded. If a match is not found, the search is extended to cover an area around the macroblock in the reference frame. Typically, this comprises a number of macroblocks as shown in Figure 4.12(b).

In practice, the various standards do not specify either the extent of the search area or a specific search strategy and instead specify only how the results of the search are to be encoded. Normally, only the contents of the *Y* matrix are used in the search and a match is said to be found if the mean of the absolute errors in all the pixel positions in the difference macroblock is less than a given threshold. Hence, using a particular strategy, all the possible macroblocks in the selected search area in the reference frame are searched for a match and, if a close match is found, two parameters are encoded. The first is known as the **motion vector** and indicates the (x,y) offset of the macroblock being encoded and the location of the block of pixels in the reference frame which produces the (close) match. The search – and hence offset – can be either on macroblock boundaries or, as in the figure, on pixel boundaries. The motion vector is then said to be **single-pixel resolution**. The second parameter is known as the **prediction error** and comprises three matrices (one each for *Y*, C_b and C_r) each of which contains the difference values (in all the pixel locations) between those in the target macroblock and the set of pixels in the search area that produced the close match.

Since the physical area of coverage of a macroblock is small, the motion vectors can be relatively large values. Also, most moving objects are normally much larger than a single macroblock. Hence, when an object moves, multiple macroblocks are affected in a similar way. The motion vectors, therefore, are encoded using differential encoding (DE) and the resulting codewords are then Huffman encoded. The three difference matrices, however, are encoded using the same steps as for I-frames: DCT, quantization, entropy encoding. Finally, if a match cannot be found – for example if the moving object has moved out of the extended search area – the macroblock is encoded independently in the same way as the macroblocks in an I-frame.

To encode a B-frame, any motion is estimated with reference to both the immediately preceding I- or P-frame and the immediately succeeding P- or I-frame. The general scheme is shown in Figure 4.13. The motion vector and difference matrices are computed using first the preceding frame as the

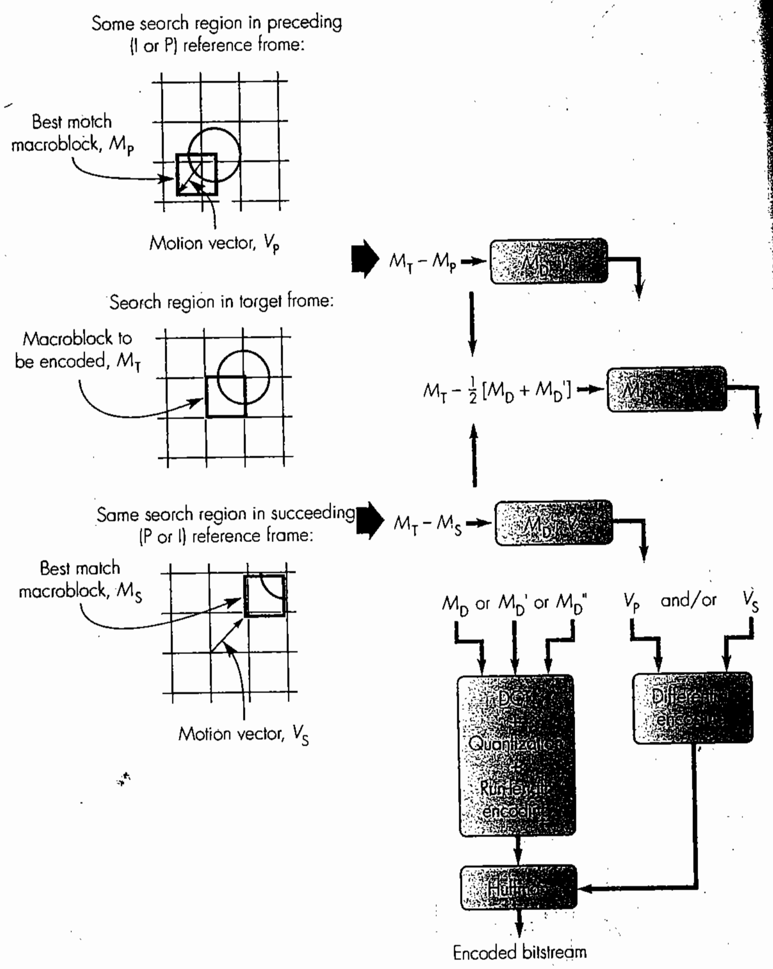

Some search region in preceding
(I or P) reference frame:
Best match macroblock, M_P
Motion vector, V_P
Search region in target frame:
Macroblock to be encoded, M_T
Same search region in succeeding
(P or I) reference frame:
Best match macroblock, M_S
Motion vector, V_S
$M_T - M_P$
$M_T - \frac{1}{2}[M_D + M_D']$
$M_T - M_S$
M_D or M_D' or M_D''
V_P and/or V_S
Quantization
Encoded bitstream

Figure 4.13 B-frame encoding procedure.

reference and then the succeeding frame as the reference. A third motion vector and set of difference matrices are then computed using the target and the mean of the two other predicted sets of values. The set with the lowest set of difference matrices is then chosen and these are encoded in the same way as for P-frames. The motion vector is then said to be to a resolution of a fraction of a pixel; for example, **half-pixel resolution**.

Implementation issues

A schematic diagram showing the essential units associated with the encoding of I-, P-, and B-frames is given in parts (a), (b) and (c) of Figure 4.14 respectively and an example format of the encoded bitstream output by the encoder is given in part (d) of the figure.

As we can deduce from part (a) of the figure, the encoding procedure used for the macroblocks that make up an I-frame is the same as that used in the JPEG standard to encode each 8 × 8 block of pixels. This was described earlier in Section 3.4.5 and, as we can see, the procedure involves each macroblock being encoded using the three steps: forward DCT, quantization, and entropy encoding. Hence assuming four blocks for luminance and two for chrominance, each macroblock would require six 8 × 8 pixel blocks to be encoded.

In the case of P-frames, the encoding of each macroblock is dependent on the output of the motion estimation unit which, in turn, depends on the contents of the macroblock being encoded and the contents of the macroblock in the search area of the reference frame that produces the closest match to that being encoded. There are three possibilities:

(1) If the two contents are the same, only the address of the macroblock in the reference frame is encoded.

(2) If the two contents are very close, both the motion vector and the difference matrices associated with the macroblock in the reference frame are encoded.

(3) If no close match is found, then the target macroblock is encoded in the same way as a macroblock in an I-frame.

As we can see in Figure 4.14(b), in order to carry out its role, the motion estimation unit containing the search logic, utilizes a copy of the (uncoded) reference frame. This is obtained by taking the computed difference values – between the frame currently being compressed (the target frame) and the current reference frame – and decompressing them using the dequantize (DQ) plus inverse DCT (IDCT) blocks. After the complete target frame has been compressed, the related set of difference values are used to update the current reference frame contents ready to encode the next (target) frame. The same procedure is followed for encoding B-frames except both the preceding (reference) frame and the succeeding frame to the target frame are involved.

As we can see from the above, for each macroblock, it is necessary to identify the type of encoding that has been used. This is the role of the **formatter** and a typical format that is used to encode the macroblocks in each frame is shown in part (d) of the figure. The *type* field indicates the type of frame being encoded – I-, P-, or B- – and the *address* identifies the location of the macroblock in the frame. The *quantization value* is the threshold value

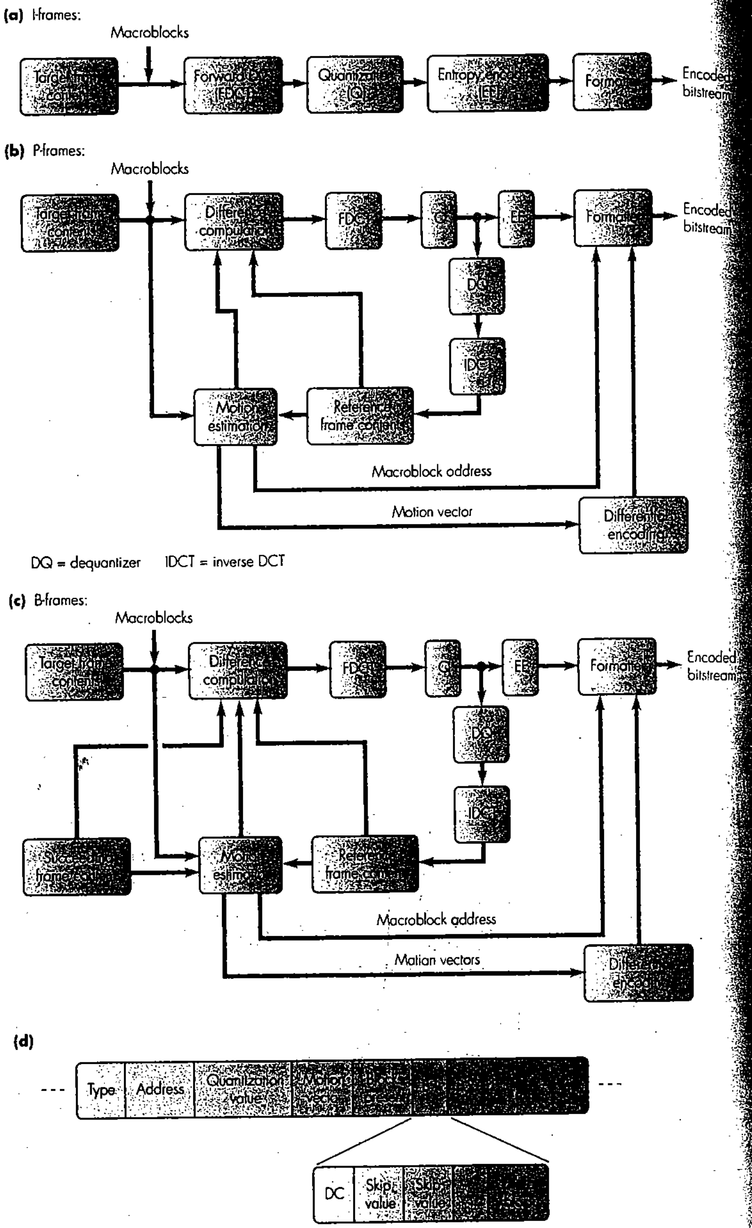

(a) I-frames:
Macroblocks
Target frame contents
Forward DCT (FDCT)
Quantization (Q)
Entropy encoder (EE)
Formatter
Encoded bitstream
(b) P-frames:
Macroblocks
Target frame contents
Difference computation
FDCT
Q
EE
Formatter
Encoded bitstream
DQ
IDCT
Motion estimation
Reference frame contents
Macroblock address
Motion vector
Difference encoding
DQ = dequantizer
IDCT = inverse DCT
(c) B-frames:
Macroblocks
Target frame contents
Difference computation
FDCT
Q
EE
Formatter
Encoded bitstream
DQ
IDCT
Succeeding frame contents
Motion estimation
Reference frame contents
Macroblock address
Motion vectors
Difference encoding
(d)
Type
Address
Quantization value
Motion vector
Blocks present
DC
Skip, value
Skip, value

Figure 4.14 Implementation schematics: (a) I-frames; (b) P-frames; (c) B-frames; (d) example macroblock encoded bitstream format.

that has been used to quantize all the DCT coefficients in the macroblock and *motion vector* is the encoded vector if one is present. The *blocks present* indicates which of the six 8 × 8 pixel blocks that make up the macroblock are present – if any – and, for those present, the JPEG-encoded DCT coefficients for each block. As we can deduce from this, the amount of information output by the encoder varies and depends on the complexity of the source video. Hence a basic video encoder of the type shown in Figure 4.14 generates an encoded bitstream that has a variable bit rate.

The decoding of the received bitstream is simpler (and hence faster) than the encoding operation since the time-consuming motion estimation processing is not required. As the encoded bitstream is received and decoded, each new frame is assembled a macroblock at a time. Decoding the macroblocks of an I-frame is the same as decoding the blocks in a JPEG-encoded image. To decode a P-frame, the decoder retains a copy of the immediately preceding (decoded) I- or P- frame in a buffer and uses this, together with the received encoded information relating to each macroblock, to build the Y, C_b, and C_r matrices for the new frame in a second buffer. With uncoded macroblocks, the macroblock's address is used to locate the corresponding macroblock in the previous frame and its contents are then transferred into the second buffer unchanged. With fully-encoded macroblocks, these are decoded directly and the contents transferred to the buffer. With macroblocks containing a motion vector and a set of difference matrices, then these are used, together with the related set of matrices in the first buffer, to determine the new values for the macroblock in the second buffer. The decoding of B-frames is similar except three buffers are used: one containing the decoded contents of the preceding I- or P-frame, the second the decoded contents of the succeeding P- or I-frame, and the third to hold the frame being assembled.

Performance

The compression ratio for I-frames – and hence all intracoded frames – is similar to that obtained with JPEG and, for video frames, typically is in the region of 10:1 through 20:1 depending on the complexity of the frame contents. The compression ratio of both P- and B- frames is higher and depends on the search method used. However, typical figures are in the region of 20:1 through 30:1 for P-frames and 30:1 through 50:1 for B-frames.

4.3.2 H.261

The H.261 video compression standard has been defined by the ITU-T for the provision of video telephony and videoconferencing services over an integrated services digital network (ISDN). Hence, as we described earlier in Section 1.3.4, it is assumed that the network offers transmission channels of multiples of 64 kbps. The standard is also known, therefore, as $p \times 64$ where p can be 1 through 30. The digitization format used is either the common intermediate format (CIF) or the quarter CIF (QCIF). Normally, the CIF used for videoconferencing and the QCIF for video telephony, both of whic we described in Section 2.6.2. However, because each frame is divided in macroblocks of 16 × 16 pixels for compression, the horizontal resolution reduced from 360 to 352 pixels to produce an integral number of 22 macr blocks. Hence, since both formats use subsampling of the two chrominanc signals at half the rate used for the luminance signal, the spatial resolution o each format is:

$$\text{CIF: } Y = 352 \times 288, \quad C_b = C_r = 176 \times 144$$

$$\text{QCIF: } Y = 176 \times 144, \quad C_b = C_r = 88 \times 72$$

Progressive (non-interlaced) scanning is used with a frame refresh rate o 30 fps for the CIF and either 15 or 7.5 fps for the QCIF.

Just I- and P- frames are used in H.261 with three P-frames between each pair of I-frames. The encoding of each of the six 8 × 8 pixel blocks that make up each macroblock in both I- and P-frames – 4 blocks for Y and one each for C_b and C_r – is carried out using the procedures described in the last section. The format of each encoded macroblock is shown in outline in Figure 4.15(a) and the format of each complete frame is shown in part (b) of the figure.

As we described in the last section, each macroblock has an *address* associated with it for identification purposes and the *type* field indicates whether the macroblock has been encoded independently – intracoded – or with reference to a macroblock in a preceding frame – intercoded. The *quantization value* is the threshold value that has been used to quantize all the DCT coefficients in the macroblock and *motion vector* is the encoded vector if one is present. The *coded block pattern* indicates which of the six 8 × 8 pixel blocks that make up the macroblock are present – if any – and, for those present, the JPEG-encoded DCT coefficients are given in each block.

The start of each new (encoded) video frame/picture is indicated by the *picture start code.* This is followed by a *temporal reference* field which is a timestamp to enable the decoder to synchronize each video block with a associated audio block containing the same time stamp. The *picture type* fiel indicates whether the frame is an I- or P-frame. Although the encoding ope ation is carried out on individual macroblocks, a larger data structure known as a *group of (macro) blocks (GOB)* is also defined. As we show in Figure 4.15(c) this is a matrix of 11 × 3 macroblocks, the size of which has been chosen so that both the CIF and QCIF comprise an integral number of GOBs – 12 in the case of the CIF (2 × 6) and 3 in the case of the QCIF (1 × 3) which allows interworking between the two formats. At the head of each GOB is a unique *start code* which is chosen so that no valid sequence of variable-length codewords from the table of codewords used in the entropy encoding stage can produce the same code. In the event of a transmission error affecting a GOB, the decoder simply searches the received bitstream for this code which signals the start of the next GOB. For this reason the *start code* is also known as a

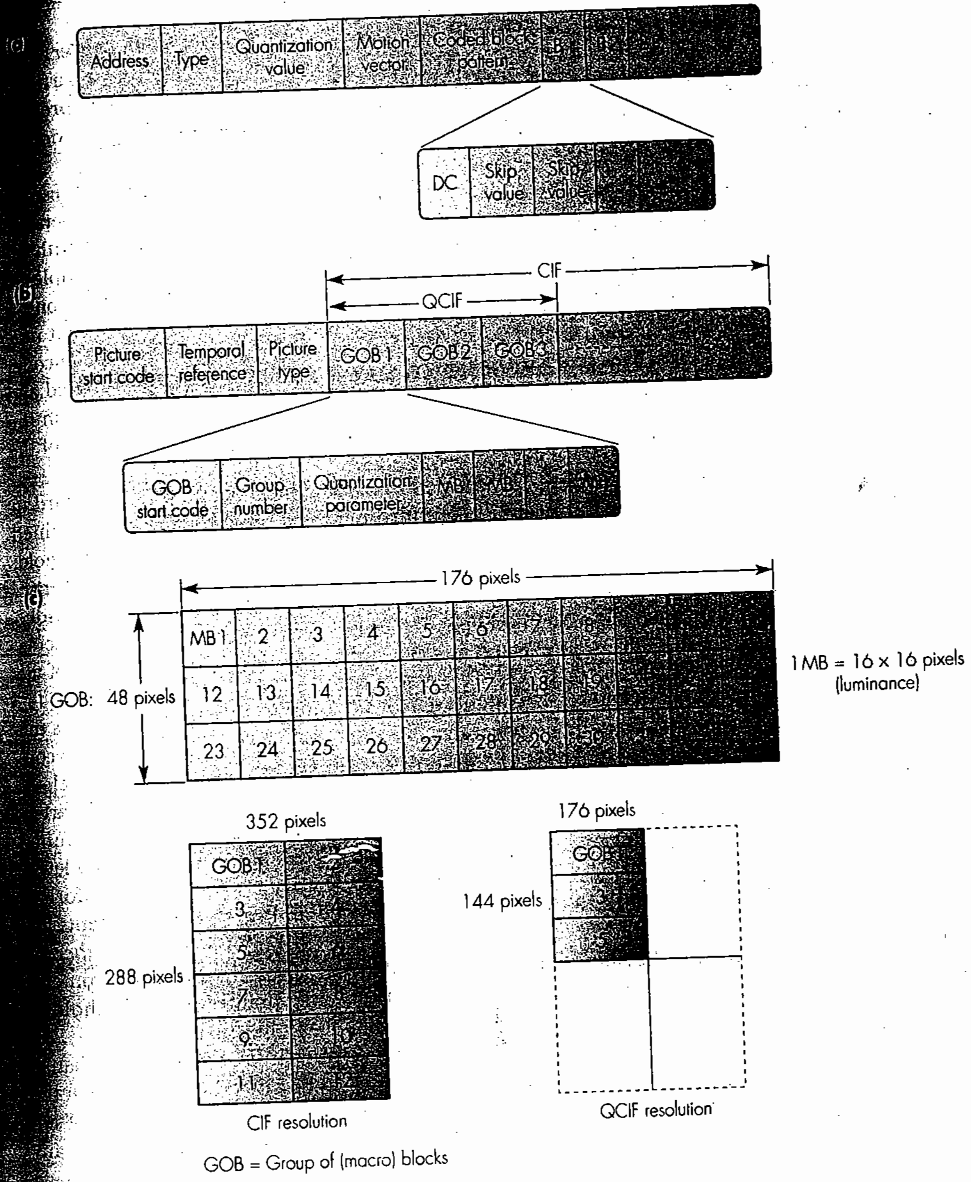

(a)
Address
Type
Quantization value
Motion vector
Coded block pattern
DC
Skip value
Skip value
(b)
CIF
QCIF
Picture start code
Temporal reference
Picture type
GOB 1
GOB 2
GOB 3
GOB start code
Group number
Quantization parameter
MB1
MB2
(c)
176 pixels
MB 1
2
3
4
5
6
7
8
12
13
14
15
16
17
18
23
24
25
26
27
28
29
1 GOB: 48 pixels
1 MB = 16 × 16 pixels (luminance)
352 pixels
288 pixels
GOB 1
3
5
7
9
11
CIF resolution
176 pixels
144 pixels
GOB 1
3
5
QCIF resolution
GOB = Group of (macro) blocks

Figure 4.15 H.261 encoding formats: (a) macroblock format; (b) frame/picture format; (c) GOB structure.

resynchronization marker. In addition, each GOB has a *group number* ass ated with it which allows for a string of GOBs to be missing from a particu frame. This may be necessary, for example, if the amount of (compress information to be transmitted is temporarily greater than the bandwidth the transmission channel.

As we indicated at the end of our earlier discussion on motion estim tion, the amount of information produced during the compression operatio varies. However, since the transmission bandwidth that is available with th target applications of the H.261 standard is fixed – 64 kbps or multiples o this – in order to optimize the use of this bandwidth, it is necessary to conve the variable bit rate produced by the basic encoder into a constant bit rat This is achieved by first passing the encoded bitstream output by the encode through a **first-in, first-out (FIFO) buffer** prior to it being transmitted an then providing a feedback path from this to the quantizer unit within th encoder. The general scheme is shown in Figure 4.16(a) and the role of th FIFO buffer is shown in Figure 4.16(b).

As you may recall from our earlier discussion of the JPEG standard i Section 3.4.5, the output bit rate produced by the encoder is determined by the quantization threshold values that are used; the higher the threshold the lower the accuracy and hence the lower is the output bit rate. Hence since the same compression technique is used for macroblocks in video encoders, it is possible to obtain a constant output bit rate from the encoder by dynamically varying the quantization threshold used. This is the role of the FIFO buffer.

As the name implies, the order of the output from a FIFO buffer is the same as that on input. However, since the output rate from the buffer is constant – determined by the (constant) bit rate of the transmission channel – if the input rate temporarily exceeds the output rate then the buffer will start to fill. Conversely, if the input rate falls below the output rate then the buffer contents will decrease. In order to exploit this property, two threshold levels are defined: the *low threshold* and the *high threshold*. The amount of information in the buffer is continuously monitored and, should the contents fall below the *low threshold*, then the quantization threshold is reduced – thereby increasing the output rate from the encoder. Conversely, should the contents increase beyond the *high threshold*, then the quantization threshold is increased in order to reduce the output rate from the encoder.

Normally, the control procedure operates at the GOB level rather than at the macroblock level. Hence, should the high threshold be reached, first the quantization value associated with the GOB is increased and, if this is not sufficient, GOBs are dropped until the overload subsides. Of course, any adjustments to the quantization threshold values that are made must be made also to those used in the matching dequantizer. In addition, the standard also allows complete frames to be missing in order to match the frame rate to the level of transmission bandwidth that is available.

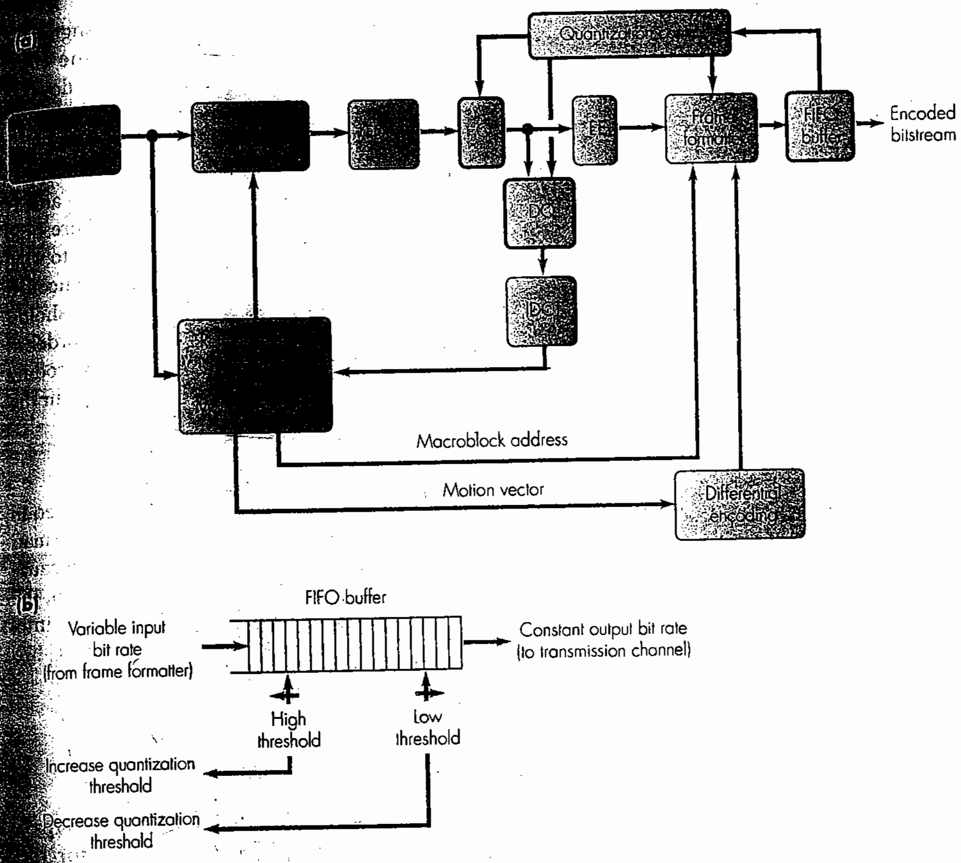

(a)
Frame formatter
FIFO buffer
Encoded bitstream
DQ
Macroblock address
Motion vector
Differential encoding
(b)
FIFO buffer
Variable input bit rate (from frame formatter)
Constant output bit rate (to transmission channel)
High threshold
Low threshold
Increase quantization threshold
Decrease quantization threshold

Figure 4.16 H.261 video encoder principles: (a) implementation schematic; (b) FIFO buffer operation.

4.3.3 H.263

The H.263 video compression standard has been defined by the ITU-T for use in a range of video applications over wireless and public switched telephone networks (PSTNs). The applications include video telephony, videoconferencing, security surveillance, interactive games playing, and so on, all of which require the output of the video encoder to be transmitted across the network connection in real time as it is output by the encoder. As we described in Section 1.3.1, the access circuit to the PSTN operates in an analog mode and to transmit a digital signal over these circuits, requires a modem. Typical maximum bit rates over switched connections range f 28.8 kbps through to 56 kbps and hence the requirement of the vi encoder is to compress the video associated with these applications dow these very low bit rates.

The basic structure of the H.263 video encoder is based on that use the H.261 standard. At bit rates lower than 64 kbps, however, the H.2 encoder gives a relatively poor picture quality. Since it uses only I- and frames, at low bit rates it has to revert to using a high quantization thresh and a relatively low frame rate. The high quantization threshold leads to w are known as **blocking artifacts** which are caused by the macroblocks encod using high thresholds differing from those quantized using lower thresho The use of a low frame rate can also lead to very jerky movements. In orde minimize these effects, the H.263 standard has a number of advanced cod options compared with those used in an H.261 encoder. We shall limit our di cussion of H.263 to an overview of a collection of these options.

Digitization formats

We described the various digitization formats associated with digital video Section 2.6.2. In the case of the H.263 standard, the two mandatory form are the QCIF and the sub-QCIF (S-QCIF). As with H.261, however, becau each frame is divided into macroblocks of 16 × 16 pixels for compression, th horizontal resolution is reduced from 180 to 176 pixels to produce an int gral number of (11) macroblocks. Hence, since subsampling of the tw chrominance signals is used, the two alternative spatial resolutions are:

QCIF: $Y = 176 \times 144,\quad C_b = C_r = 88 \times 72$

S-QCIF: $Y = 128 \times 96,\quad C_b = C_r = 64 \times 68$

Progressive scanning is used with a frame refresh rate of either 15 or 7.5 fp

In practice, the support of both formats is mandatory only for th decoder, and the encoder need support only one of them. As you may re from Section 4.3.1, the motion estimation unit is not required in the deco and hence is less expensive to implement than the encoder. The additio cost of having two alternative decoders is, therefore only small. However having a choice for the encoder, this means that either a simple – and he low-cost – encoder design based on the S-QCIF can be used for applica such as games playing, or a more sophisticated design based on the QCIF c be used for applications such as videoconferencing. The decoder can be same in both cases as it supports both formats.

Frame types

In order to obtain the higher levels of compression that are needed, th H.263 standard uses I-, P-, and B-frames. Also, in order to use as high a fram rate as possible, optionally, neighboring pairs of P- and B-frames can b encoded as a single entity. As we described in Section 4.3.1, the resultin

encoded frame is known as a PB-frame and, because of the much reduced encoding overheads that are required, its use enables a higher frame rate to be used with a given transmission channel.

As we showed earlier in Figure 4.11(c), a PB-frame comprises a B-frame and the immediately succeeding P-frame. The encoded information for the corresponding macroblock in both these frames is interleaved, with the information for the P-frame preceding that of the B-frame. Hence at the decoder, as the encoded information is received, the macroblock for the P-frame is reconstructed first using the received information relating to the P-macroblock and the retained contents of the preceding P-frame. The contents of the reconstructed P-macroblock are then used, together with the received encoded information relating to the macroblock in the corresponding B-frame and the retained contents of the preceding P-frame, to bidirectionally predict the decoded contents of the B-macroblock. Then, when the decoding of both frames is complete, the B-frame is decoded first followed by the P-frame.

Unrestricted motion vectors

As we showed earlier in Figures 4.12 and 4.13, the motion vectors associated with predicted macroblocks are normally restricted to a defined area in the reference frame around the location in the target frame of the macroblock being encoded. In addition, the search area is restricted to the edge of the frame. This means that should a small portion of a potential close-match macroblock fall outside of the frame boundary, then the target macroblock is automatically encoded as for an I-frame. This occurs even though the portion of the macroblock within the frame area is a close match.

To overcome this limitation, in the unrestricted motion vector mode, for those pixels of a potential close-match macroblock that fall outside of the frame boundary, the edge pixels themselves are used instead and, should the resulting macroblock produce a close match, then the motion vector, if necessary, is allowed to point outside of the frame area. In practice, with the small digitized frame formats that are used with the H.263 standard, this has been found to give a significant improvement in the level of compression obtained.

Error resilience

As we indicated earlier, the target network for the H.263 standard is a wireless network or a PSTN. With this type of network there is a relatively high probability that transmission (bit) errors will be present in the bitstream received by the decoder. Normally, such errors are characterized by periods when a string of error-free frames is received followed by a short burst of errors which, typically, corrupt a string of macroblocks within a frame. In practice, it is not possible to identify the specific macroblocks that are corrupted but rather that the related group of (macro)blocks (GOB) contains one or more macroblocks that are in error. Also, as we can deduce from the frame sequences we showed earlier in Figure 4.11, because the contents of many frames are predicted from information in other frames, it is highly probable that the same GOB in each of the following frames that are derived from the GOB in error will contain errors. This means that when an error in a GOB occurs, the error will persist for a number of frames, hence making the error more apparent to the viewers.

As we explained in Section 4.3.2, when an error in a GOB is detected, the decoder skips the remaining macroblocks in the affected GOB and searches for the unique resynchronization marker (start code) at the head of the next GOB. It then recommences decoding from the start of this GOB. In order to mask the error from the viewer, an **error concealment scheme** is incorporated into the decoder. For example, a common approach is to use the contents of the corresponding GOB from the preceding (decoded) frame.

In addition, since a PSTN provides only a relatively low bit rate transmission channel, to conserve bandwidth, intracoded (I) frames are inserted at relatively infrequent intervals. Hence in applications such as video telephony in which the video and audio are being transmitted in real time, the lack of I-frames has the effect that errors within a GOB may propagate to other regions of the frame due to the resulting errors in the motion estimation vectors and motion compensation information. With digitization formats such as the QCIF (which has only three GOBs per frame) the resulting effect can be very annoying to the viewer. A typical effect is shown in diagrammatic form in Figure 4.17(a) and, as we can see, although the initial error occurs in one GOB position, it rapidly spreads to other neighbouring GOBs. It is for this reason that schemes are included in the standard that aim at minimizing the effects of errors on neighbouring GOBs. The schemes include **error tracking**, **independent segment decoding**, and **reference picture selection**. We shall discuss the principle of operation of each of these schemes separately.

Error tracking With real-time applications such as video telephony, a two-way communications channel is required for the exchange of the compressed audio and video information generated by the codec in each terminal. This means that there is always a return channel from the receiving terminal back to the sending terminal and this is used in all three schemes by the decoder in order to inform the related encoder that an error in a GOB has been detected. Typically, errors are detected in a number of ways including:

- one or more out-of-range motion vectors,
- one or more invalid variable-length codewords,
- one or more out-of-range DCT coefficients,
- an excessive number of coefficients within a macroblock.

In the error tracking scheme, the encoder retains what is known as error prediction information for all the GOBs in each of the most recently transmitted frames; that is, the likely spatial and temporal effects on the macroblocks in the following frames that will result if a specific GOB in a frame is corrupted. When an error is detected, the return channel is used by the decoder to send a negative acknowledgment (NAK) message back to the encoder in the source codec containing both the frame number and the location of the GOB in the frame that is in error. The encoder then uses the error prediction information relating to this GOB to identify the macroblocks in those GOBs in later frames that are likely to be affected. It then proceeds to transmit the

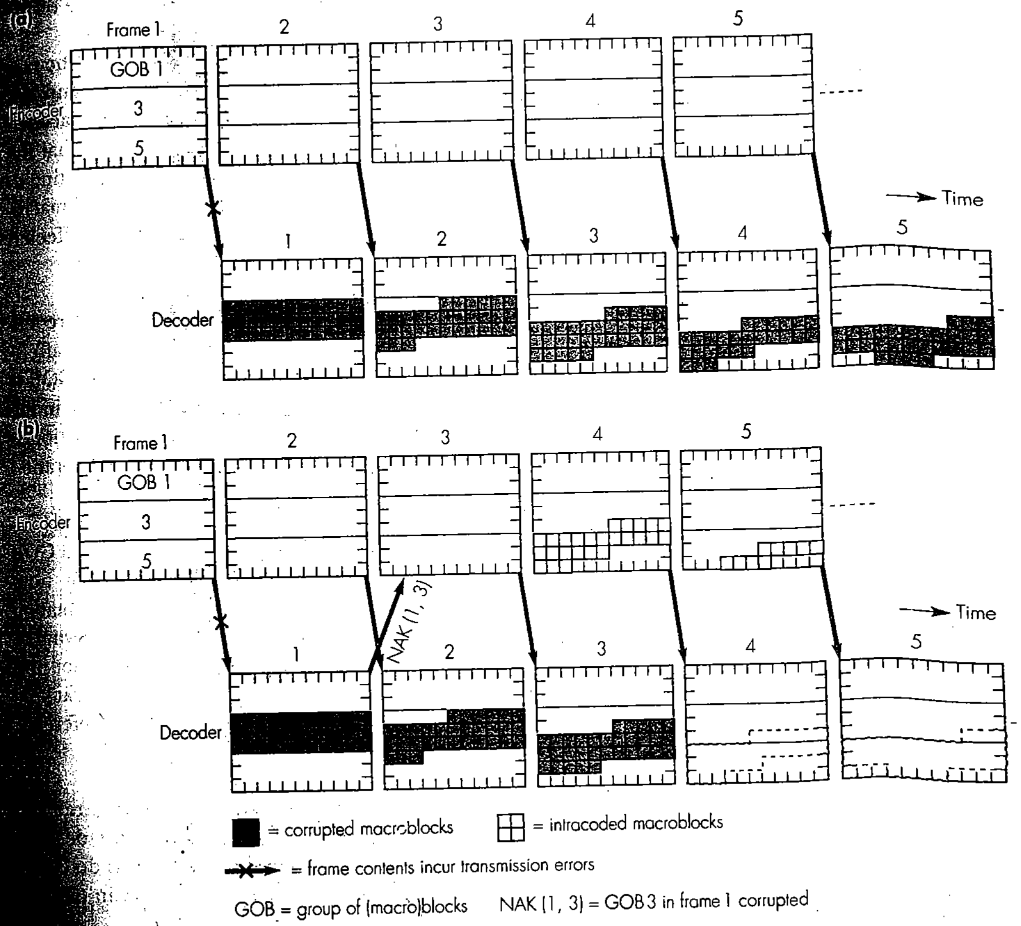


Figure 4.17 H.263 error tracking scheme: (a) example error propagation; (b) same example with error tracking applied.

macroblocks in these frames in their intracoded form. The principle of th scheme is shown in Figure 4.17(b).

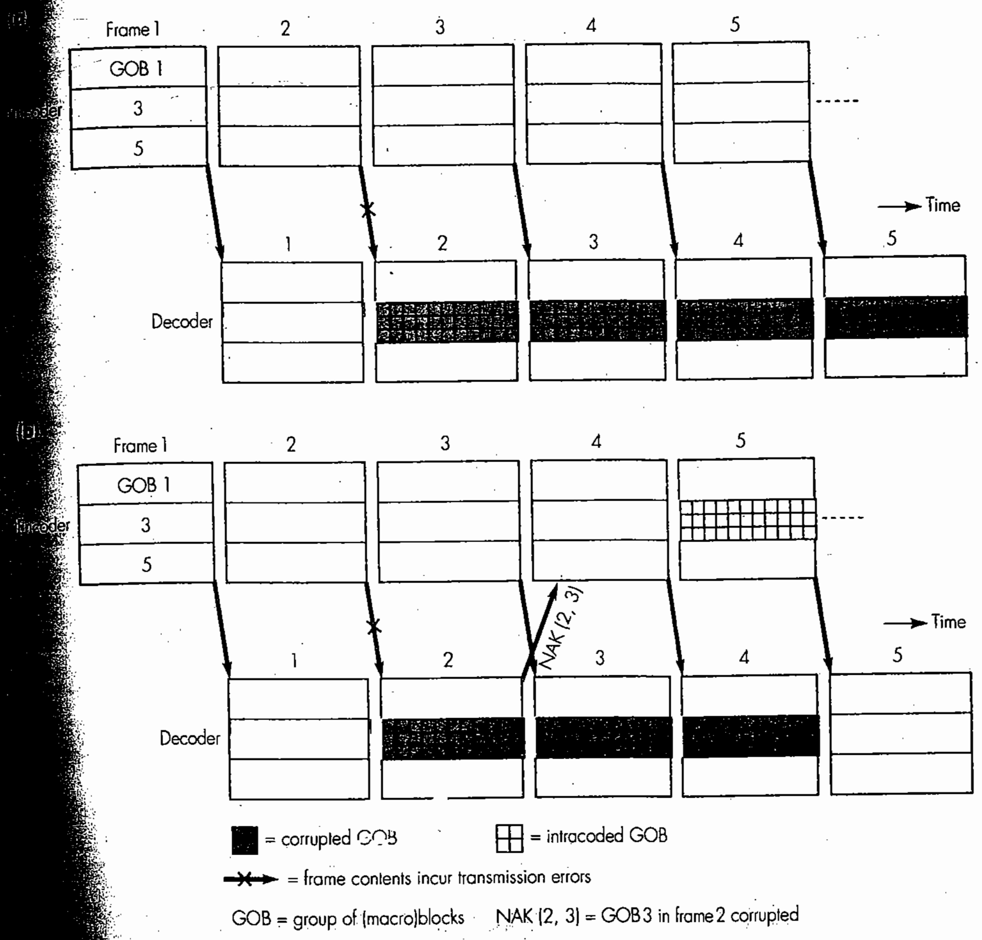
The example frame sequence shows in diagrammatic form how the co rupted macroblocks shown in part(a) of the figure are predicted and, fo each affected frame, the predicted macroblocks are sent in their intracode form. Hence in the example, on receipt of NAK(1,3), it is assumed that th encoder has predicted and retained error prediction information relating t frame 1. Since the next frame to be encoded is frame 4, the affected mac blocks are intracoded rather than predicted. Similarly for the affecte macroblocks in frame 5.

Independent segment decoding The aim of this scheme is not to overcom errors that occur within a GOB but rather to prevent these errors from affec ing neighboring GOBs in succeeding frames. To achieve this, each GOB i treated as a separate subvideo which is independent of the other GOBs in th frame. This means, therefore, that motion estimation and compensation i limited to the boundary pixels of a GOB rather than a frame. The operatio of the scheme is shown in parts (a) and (b) of Figure 4.18.

As we can see from part (a), although when an error in a GOB occurs th same GOB in each successive frame is affected – until a new intracoded GO is sent by the encoder – neighbouring GOBs are not affected. Clearly, ho ever, a limitation of this scheme is that the efficiency of the motion estimatio and compensation in the vertical direction is reduced significantly owing t the search area being limited to a single GOB. Thus, the scheme is not no mally used on its own but in conjunction with either the error trackin scheme – as we show in Figure 4.18(b) – or, more usually, with the referen picture selection scheme described below.

Reference picture selection This scheme is similar to the error track scheme inasmuch as it endeavors to stop errors propagating by the deco returning acknowledgment messages when an error in a GOB is detec The scheme can be operated in two different modes as we show in par and (b) of Figure 4.19.

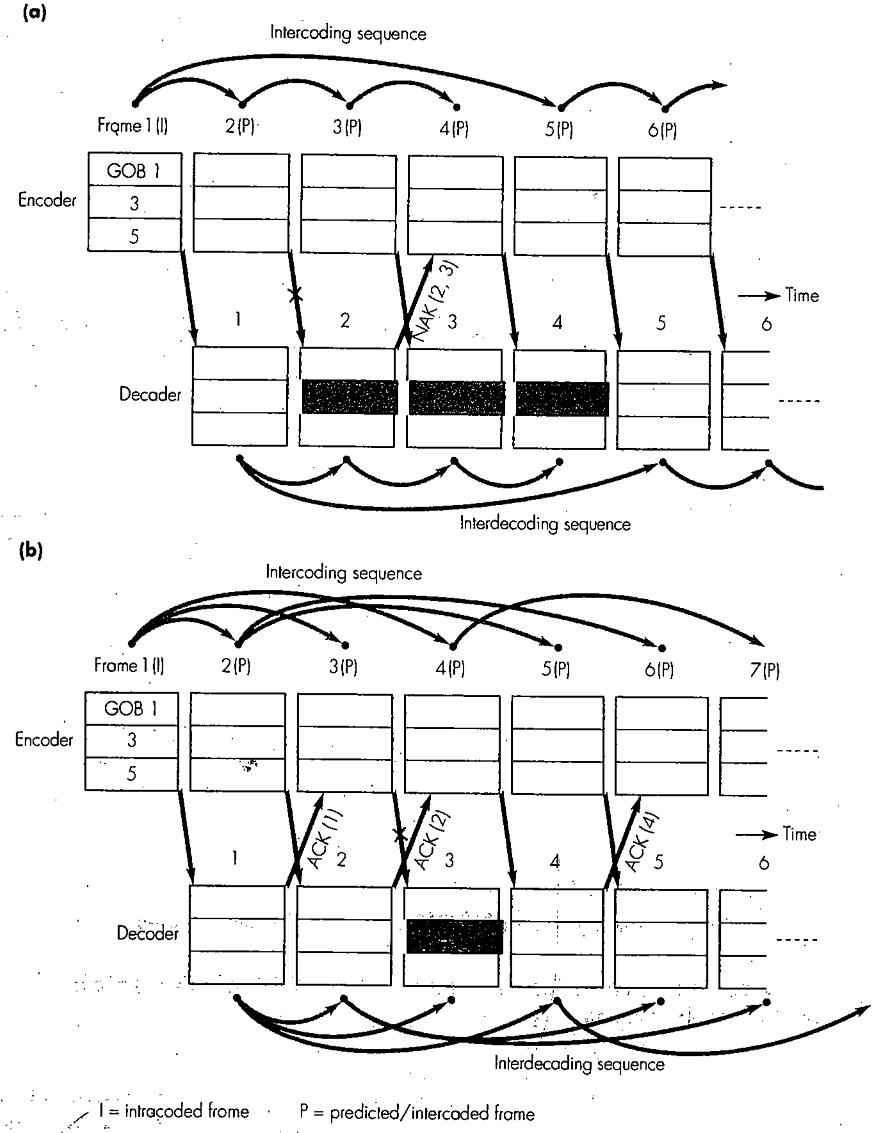
The mode of operation we show in part (a) is known as the *NAK m* and, in this mode, only GOBs in error are signaled by the decoder retur a NAK message. As we showed earlier in Figure 4.11, normally intercoded or B) frames are encoded using an intracoded (I) frame as the initial re ence frame. However, as we showed in Figure 4.14, during the encoding intercoded frames a copy of the (decoded) preceding frame is retained the encoder. With this scheme, therefore, the encoder can select any of th previously decoded frames as the reference. Hence in the example shown Figure 4.19(a), when the NAK relating to frame 2 is received, the encod selects (the decoded) GOB 3 of frame 1 as the reference to encode GOB 3 the next frame – frame 5 in this example.



4.18 Independent segment decoding: (a) effect of a GOB being corrupted; hen used with error tracking.

As we can see in the figure, with this scheme the GOB in error will propagate for a number of frames, the number being determined by the round-trip delay of the communications channel; that is, the time delay between the NAK being sent by the decoder and an intercoded frame derived from the initial I-frame being received.

The alternative mode of operation is known as the *ACK mode* since, as we show in Figure 4.19(b), with this mode all frames received without errors are



re 4.19 Reference picture selection with independent segment decoding: (a) NAK mode
ACK mode.

acknowledged by the decoder returning an ACK message. Only frames that have been acknowledged are used as reference frames. Hence, in the example, the lack of an ACK for frame 3 means that frame 2 must be used to encode frame 6 in addition to frame 5. At this point the ACK for frame 4 is received and hence the encoder then uses this to encode frame 7. The effect of using a reference frame which is distant (in time) from the frame being encoded is to reduce the encoding efficiency for the frame. Thus the ACK mode performs best when the round-trip delay of the communications channel is short and, ideally, less than the time the encoder takes to encode each frame.

4.3.4 MPEG

The **Motion Pictures Expert Group** (**MPEG**) was formed by the ISO to formulate a set of standards relating to a range of multimedia applications that involve the use of video with sound. The outcome is a set of three standards which relate to either the recording or the transmission of integrated audio and video streams. Each is targeted at a particular application domain and describes how the audio and video are compressed and integrated together. The three standards, which use different video resolutions, are:

- **MPEG-1:** This is defined in a series of documents which are all subsets of **ISO Recommendation** 11172. The video resolution is based on the source intermediate digitization format (SIF) with a resolution of up to 352 × 288 pixels. The standard is intended for the storage of VHS-quality audio and video on CD-ROM at bit rates up to 1.5 Mbps. Normally, however, higher bit rates of multiples of this are more common in order to provide faster access to the stored material.
- **MPEG-2:** This is defined in a series of documents which are all subsets of **ISO Recommendation** 13818. It is intended for the recording and transmission of studio-quality audio and video. The standard covers four levels of video resolution:
  – Low: based on the SIF digitization format with a resolution of up to 352 × 288 pixels. It is compatible with the MPEG-I standard and produces VHS-quality video. The audio is of CD quality and the target bit rate is up to 4 Mbps;
  – Main: based on the 4:2:0 digitization format with a resolution of up to 720 × 576 pixels. This produces studio-quality digital video and the audio allows for multiple CD-quality audio channels. The target bit rate is up to 15 Mbps or 20 Mbps with the 4:2:2 digitization format;
  – High 1440: based on the 4:2:0 digitization format with a resolution of 1440 × 1152 pixels. It is intended for high-definition television (HDTV) at bit rates up to 60 Mbps or 80 Mbps with the 4:2:2 format.
  – High: based on the 4:2:0 digitization format with a resolution of 1920 × 1152 pixels. It is intended for wide-screen HDTV at a bit rate of up to 80 Mbps or 100 Mbps with the 4:2:2 format.

- **MPEG-4:** Initially, this standard was concerned with a similar range of applications to those of H.263, each running over very low bit rate channels ranging from 4.8 to 64 kbps. Later its scope was expanded to embrace a wide range of interactive multimedia applications over the Internet and the various types of entertainment networks.

Unlike these three standards which are concerned with the compressio of the multimedia information associated with an application, the **MPEG** standard is concerned with describing the structure and features of th content of the (compressed) multimedia information produced by th different standards. The resulting descriptions are then used in **searc engines** to locate particular items of material that have a defined feature.

The MPEG-3 standard was to be focussed on HDTV but was not deve oped separately as the work was subsequently incorporated into MPEG-2 Hence we shall restrict our discussion to MPEG-1, the main and high levels o MPEG-2, and selected features of the MPEG-4 standard. MPEG-7 is outside o the scope of the book.

The first three MPEG standards are in three parts: video, audio, an system. The video and audio parts are concerned with the way each i compressed and how the resulting bitstreams are formatted. The system par is concerned with how the two streams are integrated together to produce a synchronized output stream. We shall discuss just the video compression part of the MPEG-1/2 standards in this chapter and the system parts in the next chapter.

4.3.5 MPEG-1

The MPEG-1 (and MPEG-2) video standard uses a similar video compression technique as H.261. As indicated, the digitization format used with MPEG-1 i the source intermediate format (SIF) which we described in Section 2.6.2 However, as with the H.261 standard, because each frame is divided int macroblocks of 16 × 16 pixels for compression, the horizontal resolution i reduced from 360 to 352 pixels to produce an integral number of (22 macroblocks. Hence, since the two chrominance signals are subsampled a half the rate of the luminance signal, the spatial resolutions for the two type of video source are:

NTSC: $Y = 352 \times 240, \quad C_b = C_r = 176 \times 120$
PAL: $Y = 352 \times 288, \quad C_b = C_r = 176 \times 144$

Also, since the application domain of MPEG-1 is for the storage of video (with audio), progressive scanning is used with a refresh rate of 30 Hz (for NTSC) and 25 Hz (for PAL).

The standard allows the use of I-frames only, I- and P-frames only, or I-, P- and B-frames, the latter being the most common. No D-frames are supported

in any of the MPEG standards and hence, in the case of MPEG-1, I-frames must be used for the various random-access functions associated with VCRs. The accepted maximum random-access time is 0.5 seconds and so this is the main factor – along with video quality – that influences the maximum separation of I-frames in the frame sequence used. Two example sequences are:

IBBPBBPBBI... and IBBPBBPBBPBBI...

the first being the original sequence proposed for use with PAL (which has a slower frame refresh rate) and the second for use with NTSC. The second standard is now used with both systems and is illustrated in Figure 4.20.

Example 4.1

An MPEG-1 system uses the frame sequence shown in Figure 4.20.

(i) Define the terms *M* and *N* and hence determine their values for the sequence shown in the figure.

(ii) Derive a suitable reordered sequence that ensures firstly, only two frames must be stored in the decoder, and secondly, the required I- and/or P-frames are available to decode each P- and B-frame as they are received.

Answer:

(i) As we described earlier in Section 4.3.1 under the subheading of "Frame types", *M* is the distance (in frames) between a P-frame and the immediately preceding I- or P- frame, and *N* is the number of frames between two successive I-frames. The latter is known as a group of pictures or GOP. Hence for the frame sequence shown in Figure 4.20, $M = 3$ and $N = 12$.

(ii) A suitable reordered frame sequence that meets the defined requirements is:

IPBBPBBPBBIBBPBB...

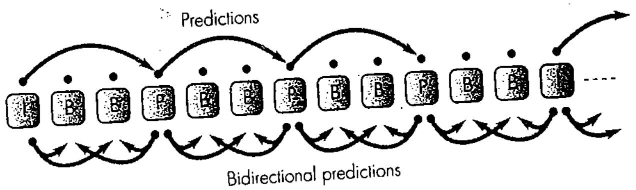


Figure 4.20 MPEG-1 example frame sequence.

The compression algorithm used in MPEG-1 is based on the H.261 standard. Hence each macroblock is made up of 16 × 16 pixels in the Y plane and 8 × 8 pixels in the C_b and C_r planes. However, there are two main differences. The first is that time-stamps (temporal references) can be inserted within a frame to enable the decoder to resynchronize more quickly in the event of one or more corrupted or missing macroblocks. The number of macroblocks between two time-stamps is known as a *slice* and a slice can comprise from 1 through to the maximum number of macroblocks in a frame. Typically, a slice is made equal to 22 which is the number of macroblocks in a line.

The second difference arises because of the introduction of B-frames which increases the time interval between I and P frames. To allow for the resulting increase in the separation of moving objects with P-frames, the search window in the reference frame is increased. Also, to improve the accuracy of the motion vectors, a finer resolution is used. Typical compression ratios vary from about 10:1 for I-frames, 20:1 for P-frames and 50:1 for B-frames.

As with the H.261 standard, the compressed bitstream produced by the video encoder is hierarchical and is shown in Figure 4.21 (a). At the top level, the complete compressed video is known as a **sequence** which, in turn, consists of a string of **groups of pictures** (GOPs) each comprising a string of I, P or B pictures/frames in the defined sequence. Each picture/frame is made up of **N slices**, each of which comprises multiple macroblocks, and so on down to an 8 × 8 pixel block. Hence in order for the decoder to decompress the received bitstream, each data structure must be clearly identified within the bitstream. The format of the bitstream is shown in Figure 4.21 (b).

mple 4.2

A digitized video is to be compressed using the MPEG-1 standard. Assuming a frame sequence of:

IBBPBBPBBPBBI...

and average compression ratios of 10:1 (I), 20:1 (P) and 50:1 (B), derive the average bit rate that is generated by the encoder for both the NTSC and PAL digitization formats.

Answer:

Frame sequence = IBBPBBPBBPBB...

Hence 1/12 of frames are I-frames, 3/12 are P-frames, and 8/12 are B-frames

and Average compression ratio = (1 × 0.1 + 3 × 0.05 + 8 × 0.02)/12

= 0.0342 or 29.24:1

NTSC frame size:

Without compression = 352 × 240 × 8 + 2 (176 × 120 × 8)

= 1.013760 Mbits per frame

With compression = 1.01376 × 1/29.24

= 34.670 kbits per frame

Hence bit rate generated at 30 fps = 1.040 Mbps

4.2 Continued

PAL frame size:

Without compression = 352 × 288 × 8 + 2 (176 × 144 × 8)

= 1.216512 Mbits per frame

With compression = 1.216512 × 1/29.24

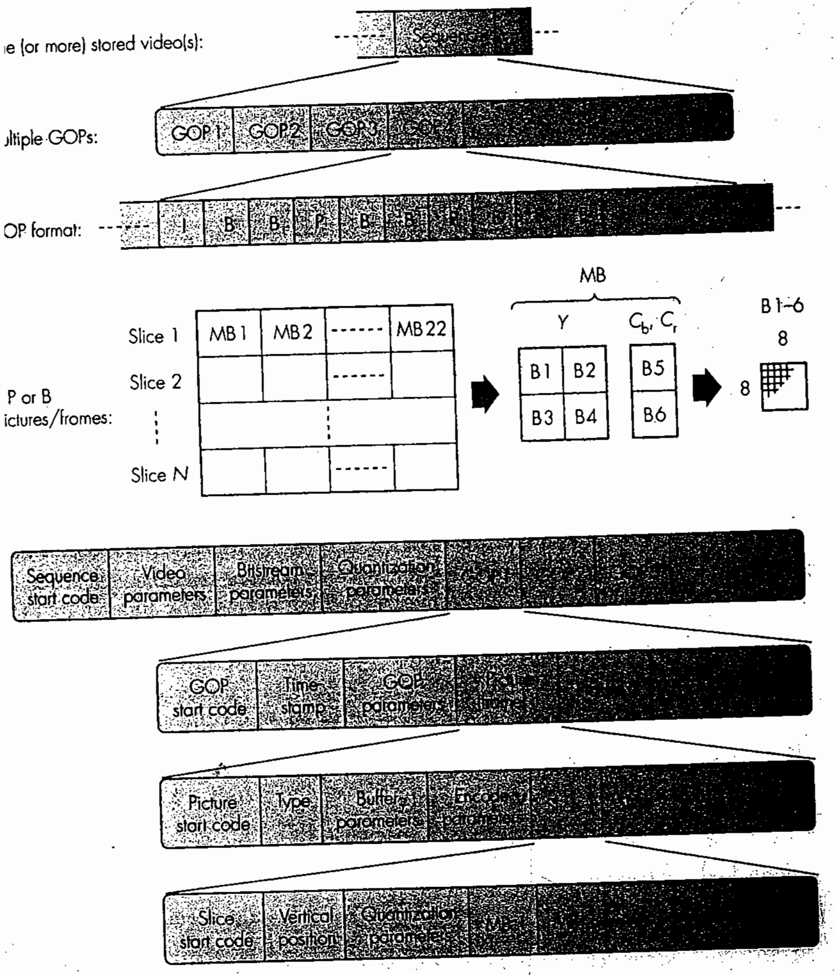
= 41.604 kbits per frame

Hence bit rate generated at 25 fps = 1.040 Mbps

Normally, allowing for packetization and multiplexing overheads, a bandwidth of 1.2 Mbps is allocated for the video. Hence, assuming a maximum bit rate of 1.5 Mbps, this leaves 300 kbps for the compressed audio stream.

The start of a sequence is indicated by a *sequence start code*. This is followed by three parameters, each of which applies to the complete video sequence. The *video parameters* specify the screen size and aspect ratio, the *bitstream parameters* indicate the bit rate and the size of the memory/frame buffers that are required, and the *quantization parameters* contain the contents of the quantization tables that are to be used for the various frame/picture types. These are followed by the encoded video stream which, as we can see, is in the form of a string of GOPs.

Each GOP (IBBP...) is separated by a *(GOP) start code* which is followed by a *time-stamp* for synchronization purposes and a *parameters* field which defines the particular sequence of frame types that are used in each GOP. This is then followed by the string of encoded pictures/frames in each GOP. Each is separated by a *picture start code* and is followed by a *type* field (I, P or B), *buffer parameters*, which indicate how full the memory buffer should be before the decoding operation should start, and *encode parameters* which indicate the resolution used for the motion vectors. This is followed by a string of slices, each comprising a string of macroblocks. Each slice is separated by a *slice start code* which is followed by a *vertical position* field, which defines the scan line the slice relates to, and a *quantization parameter* that indicates the scaling factor that applies to this slice. This is then followed by a string of macroblocks each of which is encoded in the same way as for H.261, the basic principles of which we described earlier in Section 4.3.1. As we can deduce from this, a slice is similar to the GOB used in the H.261 standard.


e (or more) stored video(s):
Sequence
ltiple GOPs:
GOP 1
GOP 2
GOP 3
GOP 4
OP format:
I
B
B
P
B
B
MB
Y
C_b, C_r
B1–6
8
8
Slice 1
MB 1
MB 2
MB 22
Slice 2
P or B
ictures/frames:
Slice N
B1
B2
B3
B4
B5
B6
Sequence start code
Video parameters
Bitstream parameters
Quantization parameters
GOP
GOP start code
Time stamp
GOP parameters
Picture start code
Type
Buffers parameters
Encode parameters
Slice start code
Vertical position
Quantization parameters
MB

21 MPEG-1 video bitstream structure: (a) composition; (b) format.

4.3.6 MPEG-2

As we indicated in Section 4.3.4, MPEG-2 supports four levels – low, main high 1440 and high – each targeted at a particular application domain. In addition, there are five **profiles** associated with each level: simple, main, spa tial resolution, quantization accuracy, and high. These have been defined so that the four levels and five profiles collectively form a two-dimensional table

which acts as a framework for all standards activities associated with MPEG-2. In this way the development of both the existing standards and any new standards can take place relative to one another. For example, at a particular level, the decoders used with a given profile will be able to decode all the lower profiles that have been defined for that level, hence making interworking between older and newer equipments possible. As indicated earlier, since the low level of MPEG-2 is compatible with MPEG-1, we shall restrict our discussion of MPEG-2 to the **main profile at the main level (MP@ML)** and the two high levels relating to HDTV.

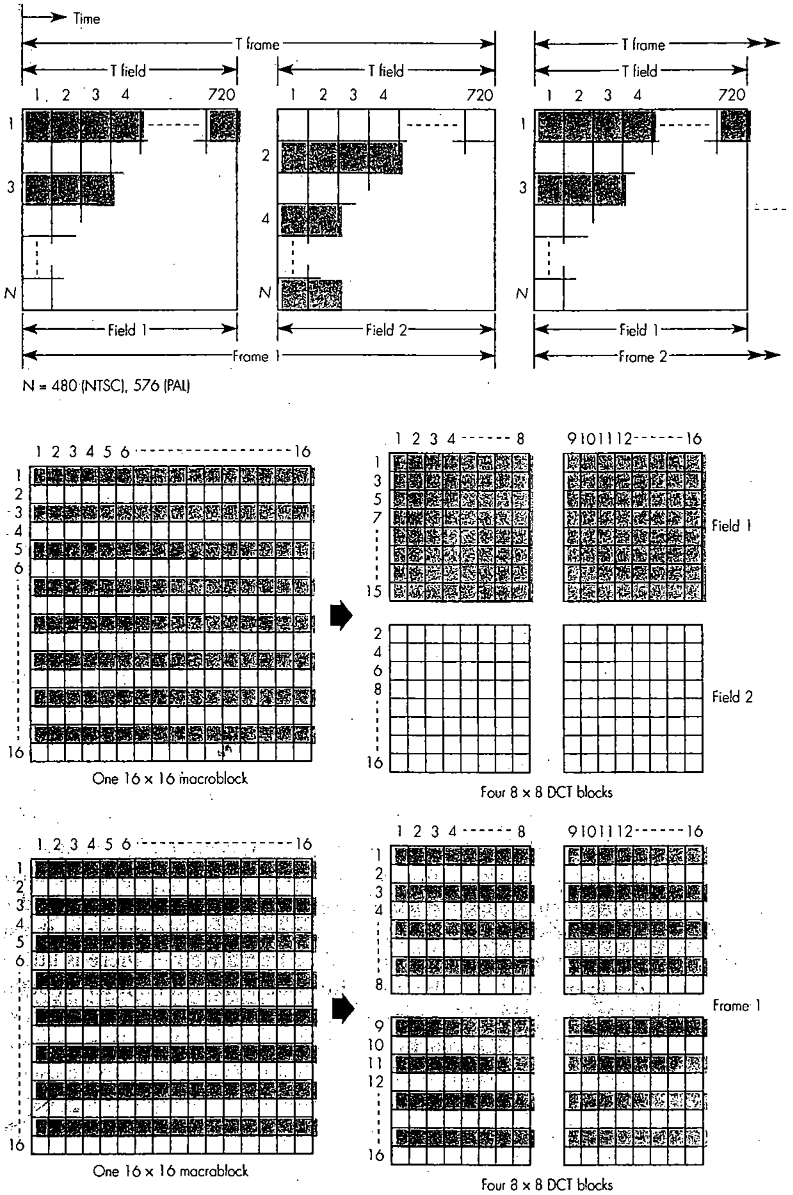
MP@ML

The target application of the MP@ML standard is for digital television broadcasting. Hence interlaced scanning is used – which we described in Section 2.6.1 – with a resulting frame refresh rate of either 30 Hz (NTSC) or 25 Hz (PAL). The 4:2:0 digitization format is used with a resolution of either 720 × 480 pixels at 30 Hz or 720 × 576 pixels at 25 Hz. The output bit rate from the system multiplexer can range from 4 Mbps through to 15 Mbps, the actual rate used being determined by the bandwidth available with the broadcast channel.

The video coding scheme is similar to that used in MPEG-1 the main difference resulting from the use of interlaced – instead of progressive – scanning. The use of interlaced scanning means that each frame is made up of two (interlaced) fields and, as we show in Figure 4.22(a), alternative lines are present in each field. Hence for I-frames, the question arises as to how the DCT blocks are derived from each macroblock. As we show in parts (b) and (c) of the figure, two alternatives are possible depending on whether the DCT blocks are derived from the lines in a field – **field mode** – or the lines in a frame – **frame mode**.

As described in Section 2.6.1, for a frame refresh rate – temporal resolution – of 30/25 Hz, the corresponding field refresh rate is 60/50 Hz. The choice of field or frame mode is thus determined by the amount of motion present in the video. If a large amount of motion is present, then it is better to perform the DCT encoding operation on the lines in a field – part (b) – since this will produce a higher compression ratio owing to the shorter time interval between successive fields. Alternatively, if there is little movement, the frame mode can be used since the longer time interval between successive (complete) frames is less important. Hence in this case, the macroblocks/DCT blocks are derived from the lines in each complete frame – part (c). In fact, the standard allows either mode to be used, the choice being determined by the type of video; for example, a live sporting event is likely to be encoded using the field mode and a studio-based program the frame mode.

In relation to the motion estimation associated with the encoding of macroblocks in P- and B-frames, three different modes are possible: field, frame, and mixed. In the field mode, the motion vector for each macroblock



2 MPEG-2 DCT block derivation with I-frames: (a) effect of interlaced scanning; ode; (c) frame mode.

is computed using the search window around the corresponding macroblock in the immediately preceding (I or P) field for both P- and B-frames and, for B-frames, the immediately succeeding (P or I) field. The motion vectors, therefore, relate to the amount of movement that has taken place in the time to scan one field. In the frame mode, a macroblock in an odd field is encoded relative to that in the preceding/succeeding odd field(s) and similarly for the macroblocks in even fields. In this case, therefore, the motion vectors relate to the amount of movement that has taken place in the time to scan two fields; that is, the time to scan a complete frame. In the mixed mode, the motion vectors for both field and frame modes are computed and the one with the smallest (mean) values is selected.

HDTV

There are three standards associated with HDTV: **advanced television (ATV)** in North America, **digital video broadcast (DVB)** in Europe, and **multiple sub-Nyquist sampling encoding** (MUSE) in Japan and the rest of Asia. All three standards, in addition to defining the digitization format and audio and video compression schemes used, also define how the resulting digital bitstreams are transmitted over the different types of broadcast network.

In addition to these standards, as with normal digital television, there is an **ITU-R HDTV specification** concerned with the standards used in television studios for the production of HDTV programs and also for the international exchange of programs. This defines a 16/9 aspect ratio with 1920 samples per line and 1152 (1080 visible) lines per frame. Currently, interlaced scanning is used with the 4:2:0 digitization format. In the future, however, it is expected that progressive scanning will be introduced with the 4:2:2 format.

The ATV standard, which was formulated by an alliance of a large number of television manufacturers, is also known as the **Grand Alliance** (or simply **GA) standard.** It includes the ITU-R HDTV specification and a second, lower-resolution format. This also uses a 16/9 aspect ratio but with a resolution of 1280 × 720. The video compression algorithm in both cases is based on the main profile at the high level (MP@HL) of MPEG-2 and the audio compression standard is based on Dolby AC-3.

The DVB HDTV standard is based on the 4/3 aspect ratio and defines a resolution of 1440 samples per line and 1152 (1080 visible) lines per frame. As we can see, this is exactly twice the resolution of the low-definition PAL digitization format of 720 × 576. The video compression algorithm is based on what is known as the **SSP@H1440 – spatially-scaleable profile at high 1440** – of MPEG-2 which is very similar to that used with MP@HL. The audio compression standard is MPEG Audio layer 2.

The MUSE standard is based on a 16/9 aspect ratio with a digitization format of 1920 samples per line and 1035 lines per frame. The video compression algorithm is similar to that used in MP@HL.

4.3.7 MPEG-4

The main application domain of the MPEG-4 standard is in relation to th
audio and video associated with interactive multimedia applications over th
Internet and the various types of entertainment networks. The standard con
tains features to enable a user not only to passively access a video sequenc
(or complete video) – using, for example, start/stop/pause commands – bu
also to access and manipulate the individual elements that make up each
scene within the sequence/video. If the accessed video is a compute
generated cartoon, for example, the user may be given the capability – by th
creator of the video – to reposition, delete, or alter the movement of the ind
vidual characters within a scene. In addition, because of its high coding
efficiency with scenes such as those associated with video telephony, the stan
dard is also used for this type of application running over low bit rat
networks such as wireless and PSTNs. For such applications, therefore, it is an
alternative to the H.263 standard.

Scene composition

The main difference between MPEG-4 and the other standards we have con
sidered is that MPEG-4 has a number of what are called **content-based**
functionalities. Before being compressed each scene is defined in the form o
a background and one or more foreground **audio-visual objects (AVOs)**
Each AVO is in turn defined in the form of one or more **video objects** and/o
audio objects; for example, a stationary car in a scene may be defined using
just a single video object while a person who is talking may be defined usin
both an audio and a video object. In a similar way, each video and audi
object may itself be defined as being made up of a number of subobjects. Fo
example, if a person's eyes and mouth are the only things that move, in orde
to exploit this, the person's face may be defined in the form of three subo
jects: one for the head and the other two for the eyes and mouth. Once th
has been done, the encoding of the background and each AVO is carried o
separately and, in the case of AVOs consisting of both audio and vide
objects, each has additional timing information relating to it to enable
receiver to synchronize the various objects and subobjects together befo
they are decoded.

Each audio and video object has a separate **object descriptor** associate
with it which allows the object – providing the creator of the audio and/o
video has provided the facility – to be manipulated by the viewer prior to
being decoded and played out. The language used to describe and mod
objects is called the **binary format for scenes (BIFS)**. This has commands
delete an object and, in the case of a video object, change its shape, appea
ance – color for example – and, if appropriate, animate the object in re
time. Audio objects have a similar set of commands, to change its volume fo
example. It is also possible to have multiple versions of an AVO, the first con
taining the base-level compressed audio and video streams and the other
various levels of enhancement detail. This type of compression is calle

scaling and allows the encoded contents of an AVO to be played out at a rate and resolution that matches those of the interactive terminal being used.

At a higher level, the composition of a scene in terms of the AVOs it contains is defined in a separate **scene descriptor**. This defines the way the various AVOs are related to each other in the context of the complete scene; for example, the position of each AVO on the display screen and the composition of each AVO in terms of the objects and subobjects it comprises.

A simple example illustrating how a frame/scene is defined in the form of a number of AVOs using content-based image coding is shown in Figure 4.23. As we can see, each video frame is segmented into a number of **video object planes (VOPs)** each of which corresponds to an AVO of interest. Hence in our simple example, the frame is shown as consisting of three VOPs: VOP 0 to represent the person approaching the car, VOP 1 the remainder of the car parked outside the house, and VOP 2 the remainder of the background. Each VOP is encoded separately based on its shape, motion, and texture.

As we show in the figure, each VOP is encapsulated within a rectangle which, in practice, is chosen so that it completely covers the related AVO using the minimum number of macroblocks. The spatial coordinates of the VOP are then relative to the coordinates of the top-left macroblock. The motion and texture of each VOP are encoded using a procedure similar, for example, to one of those we described earlier in the chapter. The resulting

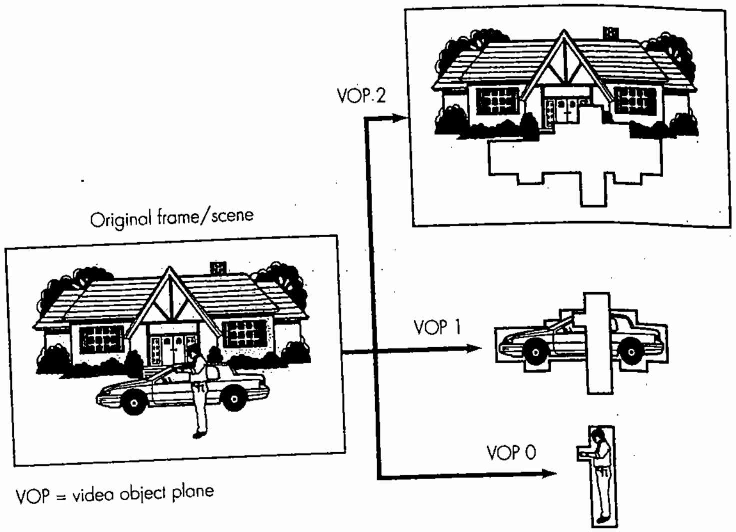


Figure 4.23 Content-based video coding principles showing how a frame/scene is defined in the form of multiple video object planes.

bitstreams are then multiplexed together for transmission together with the related object and scene descriptor information. Similarly, at the receiver, the received bitstream is first demultiplexed and the individual streams decoded. The decompressed information, together with the object and scene descriptions, are then used to create the video frame that is played out on the terminal. Typically, the latter is a multimedia PC or TV set and the audio relating to each scene is played out in time synchronism with the video.

Audio and video compression

The audio associated with an AVO is compressed using one of the algorithms we described earlier in the chapter and depends on the available bit rate of the transmission channel and the sound quality required. Examples range from G.723.1 (CELP) for interactive multimedia applications over the Internet and video telephony over wireless networks and PSTNs, through to Dolby AC-3 or MPEG Layer 2 for interactive TV applications over entertainment networks.

An overview of the structure of the encoder associated with the audio and video of a frame/scene is shown in part (a) of Figure 4.24 and the essential features of each VOP encoder are shown in part (b). As we can see, first each VOP is identified and defined and is then encoded separately. In practice, the segmentation of a video frame into various objects – from which each VOP is derived – is a difficult image-processing task. In principle, it involves identifying regions of a frame that have similar properties such as color, texture, or brightness. Each of the resulting object shapes is then bounded by a rectangle (which contains the minimum number of macroblocks) to form the related VOP. Each is then encoded based on its shape, motion, and texture. Thus any VOP which has no motion associated with it will produce a minimum of compressed information. Also, since the VOP(s) which move often occupy only a small portion of the scene/frame, the bit rate of the multiplexed videostream is much lower than that obtained with the other standards. For applications that support interactions with a particular VOP, a number of advanced coding algorithms are available to perform the shape coding function.

Transmission format

As we show in Figure 4.25, all the information relating to a frame/scene encoded in MPEG-4 is transmitted over a network in the form of a **transport stream (TS)**, consisting of a multiplexed stream of **packetized elementary streams (PESs)**. As indicated earlier, it is intended that MPEG-4 should be used with applications relating to many different types of network including the Internet, a PSTN, a wireless network, or an entertainment network – cable/satellite/terrestrial. However, as we shall expand upon in Section 5.5.2 in the next chapter, a standard 188-byte packet format has been defined for use in the various types of digital TV broadcast networks. In most cases, the MPEG-4 transport stream uses this same packet format since this helps interworking with the encoded bitstreams associated with the MPEG-1/2 standards.

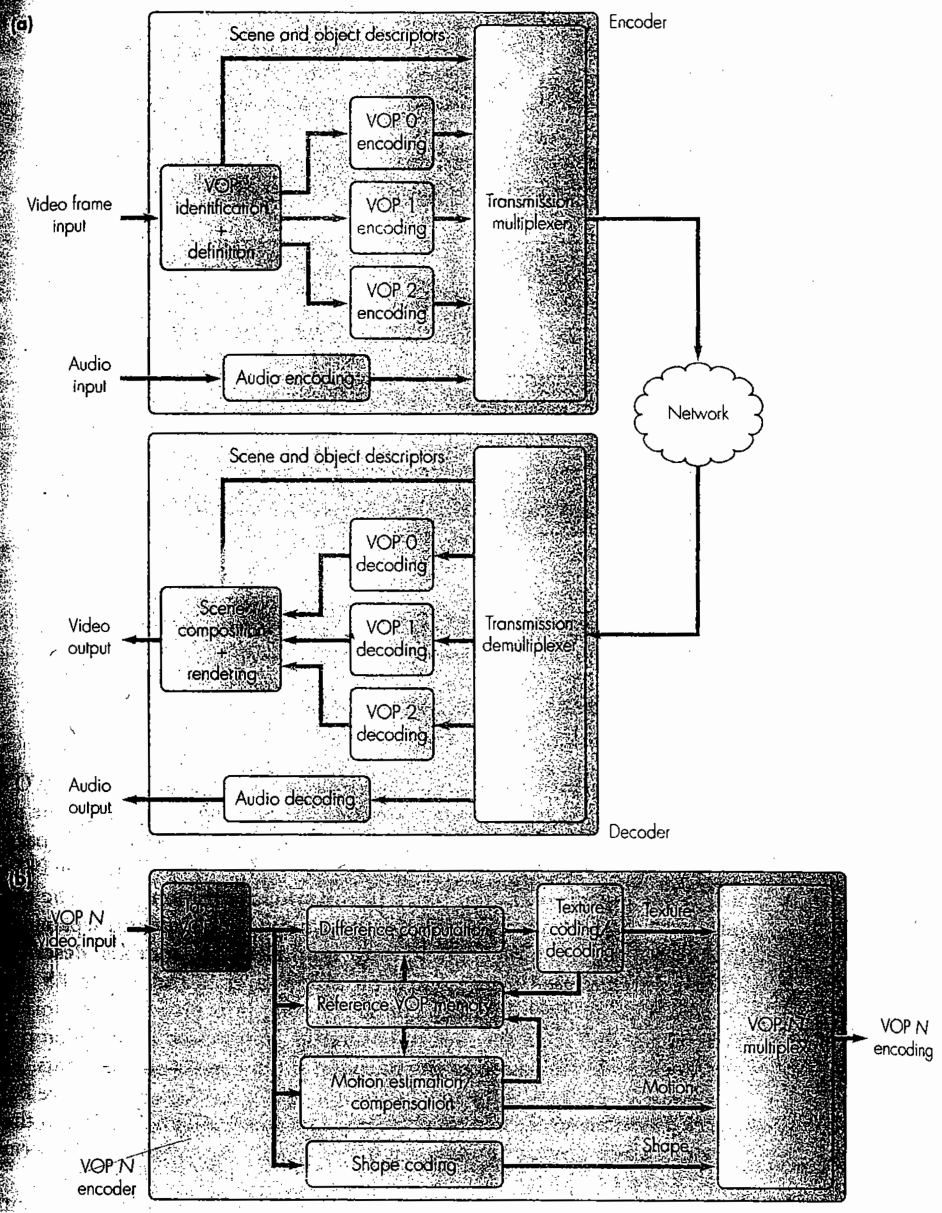
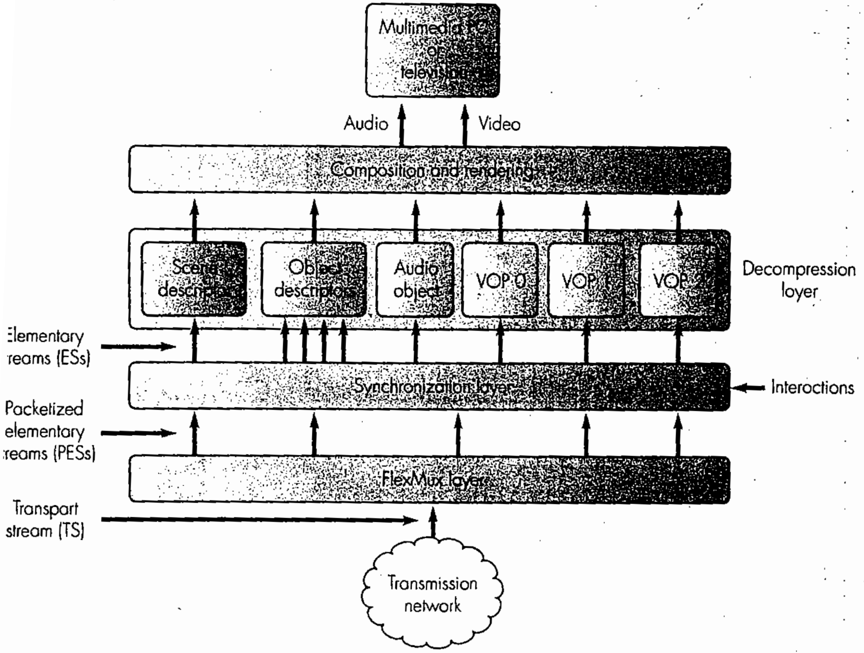

(a)
Encoder
Scene and object descriptors
VOP identification + definition
VOP 0 encoding
VOP 1 encoding
VOP 2 encoding
Transmission multiplexer
Video frame input
Audio input
Audio encoding
Network
Scene and object descriptors
VOP 0 decoding
VOP 1 decoding
VOP 2 decoding
Scene composition + rendering
Transmission demultiplexer
Video output
Audio output
Audio decoding
Decoder
(b)
VOP N video input
Difference computation
Texture coding/decoding
Texture
Reference VOP memory
Motion estimation/compensation
Motion
Shape coding
Shape
VOP N multiplex
VOP N encoding
VOP N encoder

Figure 4.24 MPEG-4 coding principles: (a) encoder/decoder schematics; (b) VOP encoder schematic.



4.25 MPEG-4 decoder schematic.

The compressed audio and video information relating to each AVO the scene is called an **elementary stream** (ES) and this is carried in the pa load field of a PES packet. Each PES packet contains a *type* field in the pac header and this is used by the **FlexMux layer** to identify and route the PES the related synchronization block in the **synchronization layer**. The co pressed audio and video associated with each AVO is carried in a separate Associated with each object descriptor is an **elementary stream descrip** **(ESD)** which is used by the synchronization layer to route each ES to related decoder. Each PES also contains timing information in the form time-stamps and these are used to ensure the compressed information rel ing to each AVO is passed to the decoder at the correct time instants. Th stream of object descriptor blocks and the scene descriptor block are bo carried in separate elementary streams and these are also passed to the related decoder blocks in time synchronism with the compression inform tion associated with the individual AVOs.

The output from the decoders associated with each AVO making up frame, together with the related object scene descriptor information, is the passed to the **composition and rendering block**. As the name implies, the

latter takes the decompressed data relating to each AVO and, together with the scene descriptor information, uses this to compose each frame ready for output to the display screen. The audio relating to each frame/scene is then played out in time synchronism with the video.

Error resilience techniques

As with the H.263 standard, a number of techniques are included in the MPEG-4 standard which make the encoding scheme more resilient to transmission errors. As we indicated earlier, these are important with transmission channels such as those associated with wireless networks and PSTNs. They include:

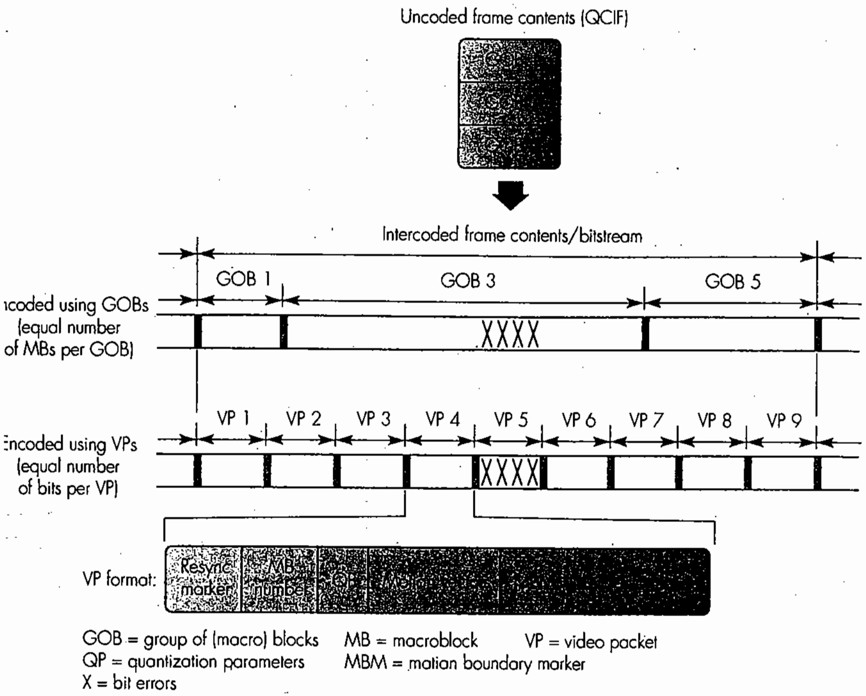
- the use of fixed-length video packets as the base-level data structure instead of GOBs;
- a new variable-length coding (VLC) scheme based on reversible VLCs.

We shall discuss the principles of each technique separately.

Video packets As we explained earlier in Sections 4.3.2 and 4.3.3 in the discussions relating to the error resilience techniques used in the H.261 and H.263 standards, resynchronization markers are inserted by the encoder at the start of each GOB to enable the decoder, on detecting an error in a GOB, to locate the end of the affected GOB and the start of the next GOB from where decoding recommences. A similar technique is used in the MPEG-1 and MPEG-2 standards at the slice level. In each case, however, when an error in a GOB/slice occurs, the complete set of macroblocks in the GOB/slice is discarded as there is no way of identifying the specific macroblocks that are corrupted.

In practice, the amount of compressed information in each GOB varies and depends on the level of activity/motion taking place in the related part of the frame. Thus, as we show in Figure 4.26(a), the number of bits in each GOB varies. In the figure, for example, it is assumed that the QCIF format is being used and that most of the activity is occurring in GOB 3 which, therefore, is much longer than the other two GOBs. Should a transmission error in a frame occur, it is highly probable that GOB 3 will be affected. This, of course, is the one we would prefer not to be affected. To overcome this effect, in MPEG-4 the compressed bitstream is divided into groups each comprising, not an equal number of macroblocks, but rather an equal number of bits. The resulting group of (compressed) bits is known as a **video packet** and, as with GOBs, these are separated by resynchronization markers.

The effect of using fixed-length video packets for the transmission of the compressed bitstream shown in Figure 4.26(a) is shown in part(b) of the figure and, as we can see, the effect is to have multiple resynchronization markers in those areas of the frame where most activity is taking place. With


Uncoded frame contents (QCIF)
Intercoded frame contents/bitstream
GOB 1
GOB 3
GOB 5
ıcoded using GOBs (equal number of MBs per GOB)
XXXX
VP 1
VP 2
VP 3
VP 4
VP 5
VP 6
VP 7
VP 8
VP 9
ncoded using VPs (equal number of bits per VP)
XXXX
VP format:
Resync marker
MB number
QP
GOB = group of (macro) blocks
MB = macroblock
VP = video packet
QP = quantization parameters
MBM = motion boundary marker
X = bit errors

4.26 MPEG-4 encoding: (a) conventional GOB approach; (b) using fixed-length video ; (c) video packet format.

this scheme, therefore, when an error occurs, only a reduced set of macroblocks are affected leaving the macroblocks in the remaining packets unaffected. In the standard, the suggested spacing of the resynchronization markers is determined by the link bit rate of the transmission channel; for example, for a bit rate of 24 kbps, the suggested spacing – and hence length of each video packet – is 512 bits.

The format of each video packet is shown in Figure 4.26(c) and, as we can see, it is similar to that of a GOB – which we showed earlier in Figure 4.15 (b) – except that after the resynchronization marker (GOB start code), the GOB number is replaced by the number of the first macroblock – relative to the complete frame – in the packet. In addition, in order to make the contents of each packet independent of the contents of other packets, the motion vectors are limited to the boundaries of the macroblocks that are in the packet.

A further refinement, as we show in the figure, is to separate the motion vectors for each macroblock in the packet from the DCT information. Normally, as we saw earlier in Figure 4.14(d), for each macroblock the encoded motion vector and DCT information are located together. As a result, the motion vectors and DCT information in a GOB are distributed throughout the complete GOB with the effect that, when an error in a GOB occurs, the complete GOB must be discarded. However, by separating these two fields by a unique bit pattern – known as the **motion boundary marker (MBM)** – it is then possible to identify separately each set of values and, if only one set is corrupted, the other set can be used.

An additional level of resilience is achieved by (optionally) including in the header of each packet a copy of the picture/frame-level parameters. These include the spatial dimensions of the picture/frame, the type of coding used (intra or inter), and temporal reference information (timestamps). Clearly, if this is corrupted, then the decoder must discard the total frame contents. Hence by replicating this information in the header of each packet – or a selected number of packets – the decoder can use a voting system to determine if errors in the frame header were present and, if errors are detected, utilize the alternative information from the head of a packet instead. In order to indicate to the decoder that frame-level information is present in the header of a packet, the encoder sets a bit in the header field known as the **header extension code (HEC) bit.**

Reversible VLCs

As we showed earlier in Figure 4.14 and described in the associated text, after compression, the resulting motion vectors and DCT information are entropy encoded prior to transmission using a table of variable-length codewords (VLCs). When using conventional Huffman VLCs of the type we described in Section 3.4.5, any bit errors that occur will cause the decoder to lose synchronization and, typically, discard the remaining codewords until it finds the next motion boundary or resynchronization marker. In order to minimize this effect, MPEG-4 uses what are known as **reversible VLCs (RVLCs).** Unlike conventional VLCs which have the prefix property, RVLCs can be decoded when read either from left-to-right or right-to-left. This means that the list of codewords making up the compressed bitstream can be identified whether the bitstream is decoded in either the forward or the reverse direction.

A simple way of producing a set of RVLCs is to first choose a set of VLCs each of which has a constant **Hamming weight**; that is, each codeword has the same number of binary 1s. The associated set of RVLCs is then produced by adding a fixed-length prefix and suffix to each of the corresponding VLCs. To illustrate this, a list of RVLCs and their derivation is shown in Figure 4.27(a). In this (simple) example, each VLC in the set has a Hamming weight of 1 and the fixed-length prefix and suffix is a single binary 1. Hence each RVLC in the resulting set of RVLCs has three binary 1s in it. As we can

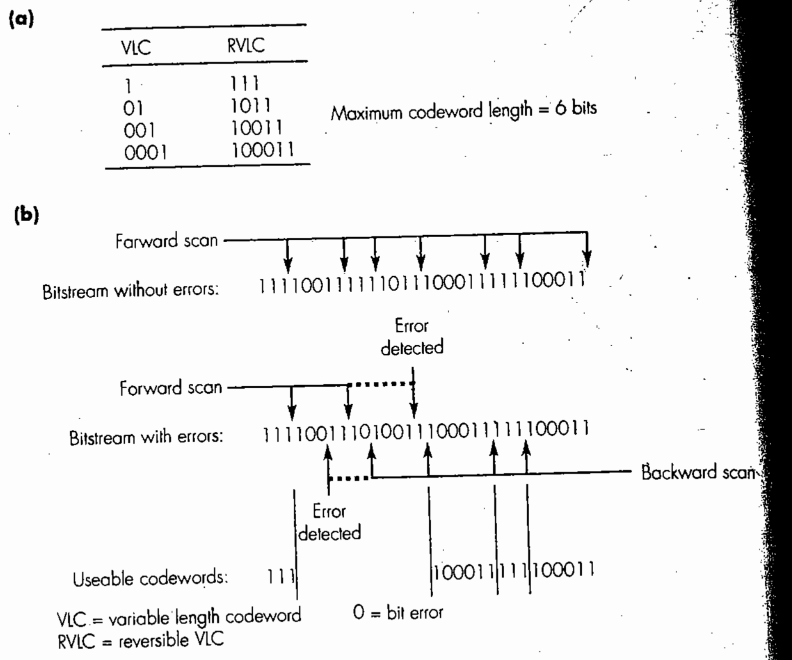

(a)
VLC
RVLC
1
01
001
0001
111
1011
10011
100011
Maximum codeword length = 6 bits
(b)
Forward scan
Bitstream without errors:
1111001111110111000111111100011
Error detected
Forward scan
Bitstream with errors:
11110011101001110001111100011
Backward scan
Error detected
Useable codewords:
111
100011
111
100011
VLC = variable length codeword
RVLC = reversible VLC
0 = bit error

Figure 4.27 Reversible VLCs: (a) example codeword set; (b) effect of transmission errors on decoding procedure.

deduce from the set of codewords, they do not have the prefix property but when decoding a string of codewords, each can be readily identified by counting the number of binary 1s and, on receipt of each third 1, the end of a codeword is signaled. Moreover, this applies whether the bitstream is scanned in the forward or the reverse direction.

To exploit this property, the decoder first scans the received bitstream in the forward direction in the normal way identifying valid codewords as it proceeds. If an invalid (corrupted) codeword is detected, it terminates the search but, instead of discarding the remaining bits, it simply skips to the next motion boundary or resynchronization marker and proceeds to scan the bitstream in the reverse direction until it encounters an invalid codeword in this direction. The general principle is shown in Figure 4.27(b) and, as we can see, when an error is present, the two scans locate the error at different points in the bitstream resulting in an overlap region. The usable codewords are then determined by ensuring that, for each direction, the position pointer of the last correctly decoded codeword is less than the corresponding error detection pointer in the opposite direction.

Summary

In this chapter we have described the underlying principles associated with both audio and video compression. In addition, we have presented an overview of a number of international standards that have been defined for use in a range of multimedia applications.

The standards relating to audio compression are divided into algorithms for the compression of speech – as used in a range of interpersonal communication applications – and algorithms for the compression of general audio. The latter are used primarily in entertainment applications.

The standards relating to speech compression are summarized in Table 4.3. They are all ITU-T Recommendations except for LPC-10 which is a military standard. They all relate to interpersonal applications (including telephony, video telephony, and videoconferencing) and each has been defined for use with a particular networking infrastructure.

Table 4.3 Summary of Speech compression standards and their applications.

Standard	Compression Technique	Compressed bit rate (kbps)	Speech Quality	Example applications
[illegible]	PCM + companding	64	Good	PSTN/ISDN telephony
[illegible]	Adaptive differential PCM (ADPCM)	32 16	Good Fair	Telephony at reduced bit rates
[illegible]	ADPCM with subband coding	64 56/48	Excellent Good	Audio conferencing
[illegible]	ADPCM with subband coding	40/32 24/16	Good Fair	General telephony at reduced bit rates
[illegible]	Linear predictive coding (LPC)	2.4/1.2	Poor	Telephony in military Networks
[illegible]	Code-excited LPC (CELP)	16	Good	Low delay/low bit rate Telephony
[illegible]	CELP	8	Good	Telephony in cellular (radio) networks
[illegible]	CELP	8	Good	Simultaneous telephony and data (fax)
[illegible]	CELP	6.3 5.3	Good Fair	Video and Internet Telephony

The general audio compression standards are summarized in Table As we can see, the majority are intended for use in various entertainme applications.

A summary of the various video compression standards that were describ is given in Table 4.5 and, as we can see, these have been defined for use i range of multimedia applications. The two ITU-T standards – H.261 and H.26 – are intended for use with various types of network including PSTNs, ISDN LANs, intranets, and the Internet. Example applications include video te phony, videoconferencing, and security surveillance. The three MPE standards – MPEG-1, 2, and 4 – are intended for use in various entertainme applications including the storage of video on CD-ROMs, digital video broa casting, and more general interactive multimedia applications. The MPEG standard is concerned with a means of defining the structure and conten the (compressed) information produced by the other standards.

Summary of compression standards for general audio.

	Compressed bit rate (kbps)	Quality	Example applications
io Layer 1	32 – 448	Hi-fi quality at 192 kbps	Digital audio cassette
Layer 2	32 – 192	Near CD at 128 kbps	Digital audio and digi video broadcasting
Layer 3	64	CD quality	CD-quality over low bi rate channels
io AC-1	512 kbps	Hi-fi quality	Radio and television satellite relays
AC-2	256 kbps	Hi-fi quality	PC sound cards
AC-3	192 kbps	Near CD quality	Digital video broad casting

Summary of video compression standards.

	Digitization format	Compressed bit rate	Example applications
[illegible]	CIF/QCIF	× 64 kbps	Video telephony/conferencing ove × 64 kbps channels (p = 1-30) and LANs
[illegible]	SQCIF/QCIF	<64 kbps	Video telephony/conferencing and security surveillance over low bi rate channels

Table 4.5 Continued

dard	Digitization format	Compressed bit rate	Example applications
[illegible]	SIF	<1.5Mbps	Storage of VHS quality video on CD-ROMs
[illegible]	SIF	<4Mbps	Recording of VHS quality video
	4:2:0	<15Mbps	Digital video broadcasting
	[illegible]	<20Mbps	
	[illegible]	<60Mbps	High definition television (4/3 aspect ratio)
	4:2:2	<80Mbps	
	4:2:0	<80Mbps	High definition television (16/9 aspect ratio)
	4:2:2	<100Mbps	
[illegible]	Various	5kbps tens Mbps	Versatile multimedia coding standard
[illegible]		–	Structure and content descriptions of compressed multimedia information search engines

Exercises

Section 4.2

With the aid of a schematic diagram, explain the operation of a basic DPCM signal encoder and decoder. Include in your explanation the source of errors that can arise.

Explain how the performance of a basic DPCM scheme can be improved by utilizing a more accurate version of the previous signal. Hence with the aid of a schematic diagram of the signal encoder and decoder, explain the principle of operation of a third-order DPCM scheme.

4.3 Explain how a basic ADPCM scheme obtains improved performance over a DPCM scheme. Give examples of the performance of a G.721-complaint codec.

4.4 With the aid of a schematic diagram of the signal encoder and decoder, explain how better sound quality – for the same bit rate – can be obtained using a subband coding ADPCM. Give examples of the bit rates used for the lower and higher subbands and state an application of this type of codec.

ate the principle of operation of a G.726-
ompliant codec and give some example
perating bit rates that are used.

xplain briefly how higher levels of compres-
on can be obtained by making the predictor
oefficients associated with ADPCM adaptive.

xplain the principles on which LPC codes are
ased. Hence, with the aid of a schematic dia-
ram of an LPC encoder and decoder, identify
e perception parameters and associated
ocal tract excitation parameters that are used.
xplain how these are used by the decoder to
enerate the output speech signal. State the
eaning of LPC-10.

xplain the main difference between an LPC
odec and a CELP codec. Include in your
xplanation the meaning of the terms "wave-
orm template" and "template codebook".

xplain the meaning of the following terms
elating to speech coders:
i) processing delay,
ii) algorithmic delay,
iii) lookahead,
iv) coder delay.
State the latter for a G.711 (PCM) and a
3.723.1 (CELP) coder and hence explain how
he coder delay often determines the suitability
of a coder for a specific application.

Explain the principles on which perceptual
coders are based and how they differ from an
LPC and CELP coder.

With the aid of a graph, show how the sensitiv-
ity of the human ear varies with frequency.
Explain clearly the dimensions of the y-axis
of your graph and, by showing two equal-
amplitude signals on your graph, explain how
one may be heard and the other not.

With the aid of a graph showing the sensitivity
of the human ear, explain the meaning of the
term "frequency masking". Illustrate on your
graph the masking effect of a loud signal on
neighboring signals.

With the aid of a diagram showing the sensitiv-
ity of the human ear, illustrate how the effect of
frequency masking varies with freque
Hence explain the term "critical bandwi
and identify how this also varies with freque
Deduce how this effect can be exploited.

4.14 With the aid of a graph, explain the meanin
the term "temporal masking". What are
implications of exploiting this effect?

4.15 In relation to the schematic diagram sho
in Figure 4.8 of an MPEG perceptual decod
explain the operation of/meaning of t
following:
(i) PCM encoder,
(ii) frequency subbands,
(iii) subband simple,
(iv) sampled segment duration,
(v) subband scaling factor,
(vi) psychoacoustic model,
(vii) quantizer,
(viii) frame formatter.

4.16 In relation to the schematic diagram shown i
Figure 4.8 of an MPEG perceptual decode
explain the operation/meaning of the following
(i) frame demultiplexer,
(ii) dequantizer,
(iii) synthesis filter bank,
(iv) PCM decoder.
What are the implications of not requiring
psychoacoustic model in the decoder?

4.17 In relation to MPEG perceptual coder
explain the three levels of processing used a
an application and typical bit rate of each.

4.18 With the aid of schematic diagrams, explai
the difference between the forward adaptiv
bit allocation mode as used with an MPEG pe
ceptual coder and the fixed bit allocatio
mode as used with a Dolby AC-1 coder.

4.19 With the aid of schematic diagrams, explain
the operation of the following two types of per-
ceptual coder:
(i) backward adaptive bit allocation (Dolby
AC-2)
(ii) hybrid backward/forward adaptive bit
allocation (Dolby AC-3).
State typical operational parameters with each
coder type and an example application of each.

4.3

plain the meaning of the following terms
ating to video compression:
moving pictures,
) MJPEG,
) motion estimation and compensation.
nclude in your explanation why the latter
used.

With the aid of example frame sequences,
explain the meaning of the following types of
compressed frame and the reasons for their use:
(i) I-frames,
(ii) P-frames,
(iii) B-frames.

4.22 Explain why I-frames are inserted into the com-
pressed output stream relatively frequently.
Hence explain the terms "group of pictures"
(GOP) and "prediction span".

4.23 Assuming the uncoded frame sequence is:

...IBBPBBPBBBBI...

explain the implications of transmitting the
compressed frames in this sequence. Hence
derive a recorded sequence that overcomes any
deficiencies.

With the aid of a frame sequence diagram,
explain the meaning of the term "PB-frame"
and why such frames are sometimes used.

Explain the application of D-frames and the
compression technique that is used with them.

In relation to the encoding procedure of I-, P-,
and B-frames, explain the meaning and use of
the following terms:
(i) macroblock,
(ii) macroblock address,
(iii) target frame,
(iv) preceding reference frame,
(v) succeeding reference frame,
(vi) motion vector,
(vii) prediction error.

4.27 With the aid of a diagram, explain how the
motion vector and prediction error are com-
puted for a P-frame. Use for example purposes
single-pixel resolution.

4.28 With the aid of diagrams, explain how the motion vectors(s) and prediction error are computed for a B-frame. Include in your explanation the meaning of the term "half-pixel resolution".

4.29 State and explain the encoding procedure used with
(i) the motion vector and
(ii) the prediction error.

4.30 In relation to the block schematic diagrams shown in Figure 4.14, explain
(i) the encoding of an I-frame,
(ii) the derivation of the contents of the reference frame used when encoding P- and B-frames,
(iii) the role of the frame formatter including the meaning of each of the fields shown in part (d) of the figure.

4.31 State the following relating to the CIF and QCIF formats of the H.261 encoding standard:
(i) the horizontal and vertical resolution in pixels
(ii) the number of macroblocks per frame
(iii) the number of GOBs per frame and their identity.

4.32 In relation to the macroblock and frame/picture formats shown in Figure 4.15, explain the role of the following fields:
(i) quantization value,
(ii) coded block pattern,
(iii) temporal reference,
(iv) start code/resyncronization marker.

4.33 With the aid of Figure 4.16, explain the role and operation of the quantization control block. Include in your explanation the use of the FIFO buffer and the associated high and low threshold levels.

4.34 Describe the following relating to the H.263 video compression standard assuming the QCIF:
(i) the number of pixels per frame,
(ii) the number of GOBs per frame,
(iii) the encoding of a PB frame,
(iv) unrestricted motion vectors.

4.35 With reference to the frame sequence diagram shown in Figure 4.17(a), explain how
(i) the corruption of one or more macroblocks within a GOB may corrupt the whole GOB

i) errors within a GOB may propagate to other neighbouring GOBs.

ith the aid of the frame sequence diagram own in Figure 4.17(b), explain the operation f the error tracking scheme. Include in your xplanation:

) how errors are detected,
i) error prediction information and its use,
ii) NAK messages and their content.

ith the aid of the frame sequence diagram hown in Figure 4.18(b), explain the operation f the independent segment decoding scheme. nclude in your explanation:

i) how errors in a GOB do not propagate to neighbouring GOBs,
i) how error tracking can be used to improve error concealment.

ith the aid of the sequence diagrams in Figures .19(a) and (b), explain the operation of the eference picture selection scheme. Include:

i) how the propagation of errors in a GOB is overcome in the NAK mode
ii) how the choice of reference frame is made in the ACK mode.

or the MPEG-1 frame sequence shown in igure 4.20, derive a suitable recorded sequence f frames that results in minimum delays in the lecoder.

State how the compression algorithm used with MPEG-1 differs from that used in the H.261 standard.

Assuming the frame sequence shown in Figure 4.21(a) and average ratios of 12:1 (I), 20:1 (P), and 40:1 (B), derive the average bit rate generated by an MPEG-1 encoder for both the NTSC and PAL digitization formats.

Explain the meaning of the following terms relating to the video bitstream structure shown in Figure 4.21:

(i) sequence,
(ii) GOP,
(iii) slice,
(iv) time-stamp,
(v) buffer and encode parameters.

4.43 With the aid of a diagram, explain the differenc between interlaced and progressive scanning.

4.44 With the aid of the examples shown in Figure 4.22, explain how the DCT blocks are derived from each macroblock in an I-frame
(i) in the field mode and
(ii) in the frame mode.
State an application for each mode.

4.45 With the aid of a diagram, explain the meaning of the following terms relating to MPEG-4 video compression standard:
(i) content-based functionality,
(ii) audio and video objects,
(iii) video object plain.

4.46 With the aid of the diagrams shown in Figure 4.24, explain the meaning of the terms:
(i) scene and object descriptions,
(ii) scene composition and rendering,
(iii) texture, motion, and shape encoding.

4.47 In relation to the schematic diagram of an MPEG-4 decoder shown in Figure 4.25, explain the role of the following:
(i) transport stream,
(ii) flexmux layer,
(iii) PES,
(iv) synchronization layer,
(v) ES,
(vi) decomposition layer,
(vii) composition and rendering layer.

4.48 With the aid of the diagrams shown in Figure 4.26, explain:
(i) the advantages of using fixed-length video packets instead of GOBs with lo bit rate wireless and PSTNs
(ii) the separation of the motion vectors and DCT information in each video packet b motion boundary marker
(iii) the replication of picture/frame-level parameters in the header of each video packet.

4.49 With the aid of the diagrams shown in Figure 4.27, explain:
(i) how reversible variable-length codewords (RVLCs) reduce the effects of transmission errors
(ii) the derivation of RVLCs
(iii) forward and reverse scans and their use.

5

Standards for multimedia communications

5.1 Introduction

In Chapter 1 we identified the different types of communication networks that are used to provide multimedia services. We also described a range of multimedia applications that use these networks and, although these are many and varied, we showed that they can be classified into one of three categories: interpersonal communications, interactive applications over the Internet, and entertainment applications.

In Chapter 2 we described the way the different types of media that are used in multimedia applications – text, images, speech, audio, and video – are represented in a digital form and we concluded that a number of the network types used can only support multimedia applications as a result of the technological advances that have taken place in the field of compression. Hence in Chapters 3 and 4 we described a selection of the different algorithms and standards that are used for the compression of the different media types.

In general, however, most of the multimedia applications we described in Chapter 1 involve not just a single type of medium but rather a number of media types that are integrated together in some way. For example, an

8

ıterprise networks

8.1 Introduction

When a person is at home, all calls relating to interpersonal and intera
applications must be made using a PSTN, an ISDN, or a cable distrib
network. The calls are then charged at a rate determined by the call du
and the distance involved. In the case of a person in a business or large e
prise, however, the majority of the calls made are to other members o
same business/enterprise and only a small percentage are to people o
of the enterprise. Hence for all but the smallest businesses, in order to r
call charges, most enterprises install their own private networks to h
those calls that are internal to the enterprise. Normally, the network
prises both a private branch exchange (PBX) and a local area network (
and, collectively, these support all the interpersonal and interactive com
nications within the enterprise.

For an enterprise that occupies just a single site/establishment –
example, a small to medium-sized company, a hospital complex, a univer
campus – the PBX and LAN at the establishment handle all internal calls a
only those calls that are external to the site are made using an appropri
public network such as a PSTN, an ISDN, or an Internet service provid

(ISP) network. For an enterprise that operates over multiple sites, however, when there is a significant proportion of intersite calls, an alternative solution is to extend the private facilities associated with each site to embrace all sites. This involves linking the sites together using high bit rate transmission lines that are leased ~~from~~ a (national) network provider. The resulting network is then known as a multisite **enterprise network** or, if the sites are located in different countries around the world, a **global enterprise network**.

Normally, leased lines are charged for on an annual rather than a per-call basis and hence this approach is only justified when the annual (public) call charges to other sites exceeds the cost of leasing the lines. An added benefit of creating a private network, however, is that it is often easier to offer more sophisticated services since the utilization of the bandwidth of the intersite leased lines is under the control of the enterprise network manager. Also, because the total network is private – apart from the transmission lines, of course – a private network is considered to be more secure than a public one.

As we explained in the last chapter, a PBX operates in a similar way to a local exchange/end office in a public network with the terminal equipment connected to the PBX using either analog or digital lines. In this chapter, we shall focus on the operation of the different types of LAN and the various approaches that are used to create multisite enterprise networks.

8.2 LANs

As we have just indicated, LANs are used to interconnect distributed communities of end systems – often referred to as **stations** in the context of LANs – including multimedia PCs, workstations, servers, and so on. Typically, these are distributed around an office, a single building, or a localized group of buildings, all of which belong to a single enterprise.

The early LANs – many of which are still in existence – operate using a shared, high bit rate, transmission medium to which all the stations are attached and the information frames relating to all calls are transmitted. To ensure the transmission bandwidth is shared fairly between all of the attached stations, a number of different medium access control (MAC) methods are used. These include **carrier-sense multiple-access with collision detection (CSMA/CD)** and **token ring**, both of which have a defined maximum number of attached stations and length of transmission medium associated with them. As we shall see, in practice the maximum distance is relatively small and hence most LANs of this type comprise multiple (LAN) **segments** that are interconnected together using either **repeaters** or devices known as **bridges** and a high bit rate (site-wide) **backbone subnetwork**. We explain the operation of Ethernet/IEEE802.3 LANs – which are based on the CSMA/CD MAC method – in Section 8.3 and token ring LANs in Section 8.4. The operation of bridges is explained in Section 8.5 and, as an example of a backbone network, the **fiber distributed interface (FDDI)** in Section 8.6.

More recently, higher bit rate versions of the older LAN types – now known as **legacy LANs** – have become available. To obtain the higher network throughputs that are required with multimedia applications, the central hubs associated with the earlier LANs have been upgraded to operate at much higher bit rates. Also, as we shall explain, the older hubs operate in a half-duplex mode and support only a single frame transfer at a time. Hence the newer hubs operate in a duplex mode and allow the frames relating to multiple calls to be transmitted concurrently. Examples include **fast Ethernet** hubs and **Ethernet switching** hubs, both of which we describe in Section 8.7.

In terms of the link layer protocol associated with LANs, the various LAN types all use a standard LC sublayer and there is a different MAC sublayer for each of the LAN types. We describe the structure and the user services offered by each sublayer in Section 8.8.

In multisite enterprise networks, the LANs associated with the different sites are interconnected together using various methods determined by the volume of intersite traffic involved. If this is relatively low, then switched ISDN connections can be used; if it is high, then high bit rate (digital) leased lines are used. Normally, these are leased from the operator of a national public circuit-switched network and are the same as those we described in Section 7.2.3. In both cases, however, a **gateway** is connected to the LAN at each site and this manages all intersite frame transfers. We explain the operation of some of the different types of technology that are used to interconnect the LANs at different sites in Section 8.9.

8.3 Ethernet/IEEE802.3

Ethernet networks – and the more recent derivative IEEE802.3 – are used extensively in technical and office environments. As we shall see, Ethernet has gone through many phases of development since its first introduction but, in general, the same basic mode of operation is still used. All frame transmissions between all the stations that are attached to the LAN take place over a shared 10 Mbps bus and the CSMA/CD MAC method is then used to share the use of the bus in an equitable way. We shall explain the principle of operation of this type of MAC method first and then other selected aspects in the following subsections.

CSMA/CD

Since all the stations are attached directly to the same cable/bus, it is said to operate in a **multiple access (MA) mode**. To transmit a block of data, the source station first encapsulates the data in a frame with the address of the destination station and its own address in the frame header and an FCS field at the tail of the frame. The bus operates in the **broadcast mode** which means that every frame transmitted is received by all the other stations that are attached to the bus. Hence as each of the other stations receives the frame, it

first checks the frame is free of errors using the FCS and, if it is, it compares the destination address in the header with its own address. If they are different, the station simply discards the frame; if they are the same, the frame contents are passed up to the LC sublayer for processing together with the address of the source station.

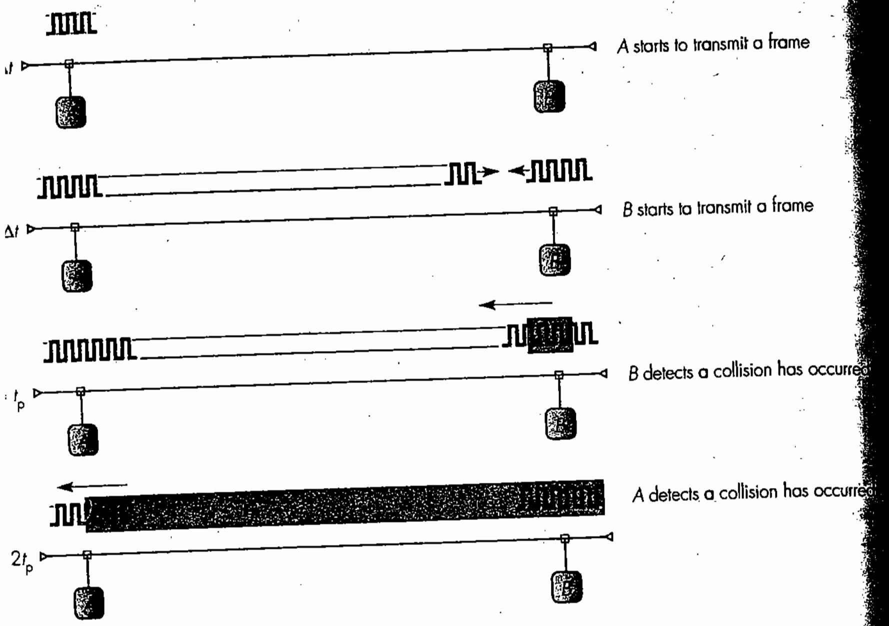
With this mode of operation, two (or more) stations may attempt to transmit a frame over the bus at the same time. Because of the broadcast mode, this will result in the contents of the two (or more) frames being corrupted and a **collision** is said to have occurred. Hence in order to reduce the possibility of a collision, prior to sending a frame the source station first determines whether a signal/frame is currently being transmitted on the bus. If a signal – known as the **carrier** – is **sensed (CS)**, the station defers its own transmission until the current frame transmission is complete and only then does it attempt to send its own frame. Even so, in the event of two (or more) stations waiting to send a frame, both will start to transmit their frame simultaneously on detecting that the transmission of the current frame is complete. When this happens, however, it is necessary for the two (or more) stations involved, to detect a collision has occurred before each has finished transmitting its own frame. In practice, because of the possibly large signal propagation delay of the bus and the high transmission bit rate used (10 Mbps), this is not as straightforward as it might seem.

A station detects that a collision has occurred by simultaneously monitoring the signal that is present on the cable all the time it is transmitting a frame. Then, if the transmitted and monitored signals are different, a collision is assumed to have occurred – **collision detected (CD)**. As we show in Figure 8.1, however, a station can experience a collision not just at the start of a frame but after it has transmitted a number of bits. The worst-case time delay – and hence maximum number of bits that have been transmitted – before detecting that a collision has taken place is known as the **collision window** and occurs when the two colliding stations are attached to opposite extremities of the bus, as we show in the figure.

In the figure, station A has determined that no transmission is in progress and hence starts to transmit a frame – part (i). As we explained in Section 6.2.8, irrespective of the bit rate being used, the first bit of the frame will take a small but finite time to propagate over the transmission medium determined by the length of the cable, l, and the signal propagation velocity, v. The maximum length of cable is set at 2.5 km. Hence, assuming a v of 2×10^8 ms^{-1}, the worst-case signal propagation delay time, T_p, going from one end of the cable to the other, is given by:

$$T_p = l/v = 2.5 \times 10^3 / 2 \times 10^8 = 12.5 \text{ microseconds}$$

Now assume that, just prior to the first bit of the frame arriving at its interface, station B determines the transmission medium is free and starts to transmit a frame – part (ii).


A starts to transmit a frame
B starts to transmit a frame
B detects a collision has occurre
A detects a collision has occurre
t
Δt
t_p
$2t_p$

= (worst-case) transmission propagation (path) delay

.1 CSMA/CD worst-case collision detection.

As we show, after *B* has transmitted just a few bits, the two signals collid part (iii) – and the collision signal then continues to propagate back to tion *A* – part (iv). Hence the worst-case time before station *A* detects th collision has occurred, $2T_p$, is 25 microseconds. In addition, as we expand upon later, in order to transmit the signal over this length of the cable is made up of five 500 m *segments*, all interconnected togethe means of four devices called **repeaters**. Each repeater introduces a delay few microseconds in order to synchronize to each new frame. Hence the worst-case time is set at 50 microseconds or, assuming a bit rate of 10 M after *A* has transmitted:

$$10 \times 10^6 \times 50 \times 10^{-6} = 500 \text{ bits}$$

A safety margin of 12 bits is then added to this and the minimum fram size is set at 512 bits or 64 bytes/octets. This is called the **slot time** (in b and ensures that station *A* will have detected a collision before it has transm ted its smallest frame. Also, to ensure that the collision signal persists f

sufficient time for it to be detected by *A*, on detecting the collision, *B* continues to send a random bit pattern for a short period. This is known as the **jam sequence** and is equal to 32 bits.

After detecting a collision, the two (or more) stations involved then wait for a further random time interval before trying to retransmit their corrupted frames. As we shall explain later, the maximum frame size including a four-byte CRC is set at 1518 bytes and hence a collision will occur if two (or more) stations create a frame to send during the time another station is currently transmitting a maximum sized frame. This is equal to a time interval of:

$$1518 \times 8/10 \times 10^6 = 1.2144 \text{ milliseconds}$$

Clearly, the probability of this occurring increases with the level of traffic (number of frames) being generated and the maximum throughput of the LAN occurs when this limit is reached. Hence if a second collision should occur when a station is trying to send a frame, this is taken as a sign that the cable is currently overloaded. To avoid further loading the cable, therefore, the time interval between trying to retransmit a frame is increased exponentially after each new attempt is made using a process known as **truncated binary exponential backoff**. The actual time is a function of the slot time. The number of slot times before the *N*th retransmission attempt is chosen as a uniformly distributed random integer R in the range $0 \leq R < 2^K$, where $K = \min(N, \text{backoff limit})$. In the standard, the **backoff limit** is set to 10.

Wiring configurations

There are a number of different types of cable that can be used with Ethernet. These include:

- 10Base2: thin-wire (0.25 inch diameter) coaxial cable with a maximum segment length of 200 m;
- 10Base5: thick-wire (0.5 inch diameter) coaxial cable with a maximum segment length of 500 m.
- 10BaseT: hub (star) topology with twisted-pair drop cables of up to 100 m;
- 10BaseF: hub (star) topology with optical fiber drop cables of up to 1.5 km.

Although different types of cable are used, they all operate using the same CSMA/CD MAC method.

At the time the first Ethernet installations were carried out, the only transmission medium available that could operate at 10 Mbps was coaxial cable. Initially, thick-wire coaxial cable was installed since this can be used in relatively long lengths of up to 500 m before the transmitted/broadcast signal needs to be repeated. As we explained in Section 6.3.1, this involves the attenuated signal received at the extremity of the cable segment being amplified and restored to its original form before it is retransmitted out onto the next cable segment. Up to five cable segments – and hence four repeaters – can be used in this way. Hence the maximum length of cable the signal propagates is 2.5 km plus 4 repeaters which is the origin of the slot-time figure used in the standard.

The disadvantage of thick-wire coax is that it is relatively difficult to bend and hence install. To overcome this, thin-wire coax was used but, because of the increased (electrical) resistance associated with it, the maximum length of cable for each segment is reduced to 200 m.

More recently, as we explained in Section 6.2.2, with the arrival of inexpensive adaptive crosstalk canceler circuits to overcome near-end crosstalk (NEXT), it is possible to obtain bit rates of tens of Mbps over twisted-pair cable of up to 100 m in length. Also, it was found that, in a vast majority of offices, the maximum length of cable used for telephony to reach each desktop from the wiring closet was less than 100 m. Hence unshielded twisted-pair (UTP) cable – as used for telephony – has rapidly become the standard fo use with Ethernet. The configuration used for each segment is shown in Figure 8.2(a).

Since the cable forms a physical bus, both thick and thin wire coaxial cable installations involve the cable passing near to each attached station. As we can see in the figure, however, with twisted-pair cable a star configuration is used with the hub located in the wiring closet and each station connected to it by means of twisted-pair drop cables. Normally, category three (CAT3) UTP cable is used as for telephony. Each cable contains four separate twisted pairs. In the case of Ethernet, just two pairs are used: one pair fo transmissions from the station to the hub and the second pair for transmissions in the reverse direction.

To emulate the broadcast mode of working associated with CSMA/CD a we show in Figure 8.2(b), the repeater electronics within the hub repeats an broadcasts out the signal received from each of the input pairs onto all of th other output pairs. Hence the signal output by any of the stations is receive by all the other stations and, as a result, the carrier sense function simp involves the MAC unit within each station determining whether a signal currently being received on its input pair. Similarly, the collision detectio function involves the station determining if a signal arrives on its input pa while it is transmitting a frame on the output pair.

Because of their mode of operation, this type of hub is called a **repea** **hub** and typical numbers of attached stations – and hence sockets – are fro 8 through to 16. Above this number multiple hubs are stacked together an are connected by repeaters or, as we shall explain in Section 8.5, bridges. I the case of repeaters, the maximum length of cable between any two statio – including the 100 m drop cables – must not exceed 1.5 km. To achieve th coverage/distance, however, normally it is necessary to use a central hub t which each twisted-pair hub is connected by means of optical fiber cables.

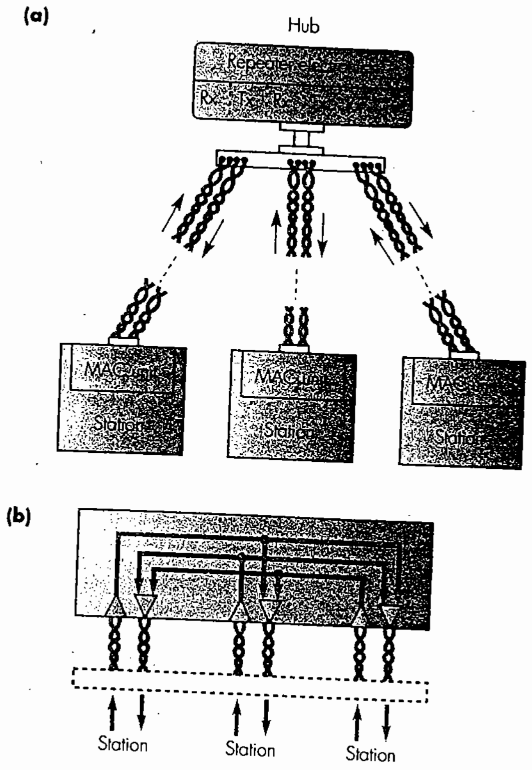
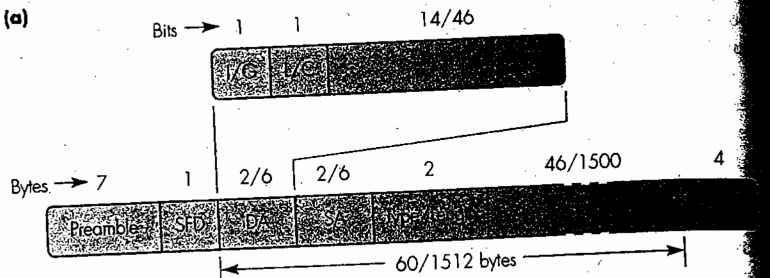


Figure 8.2 Hub configuration principles: (a) topology; (b) repeater schematic.

Frame format and operational parameters

The format of a frame and the operational parameters of a CSMA/CD network are shown in Figure 8.3. The *preamble* field is sent at the head of all frames. Its function is to allow the receiving electronics in each MAC unit and repeater to achieve bit synchronization before the actual frame contents are received. The preamble pattern is a sequence of seven bytes, each equal to the binary pattern 10101010. All frames are transmitted on the cable using Manchester encoding. Hence, as we explained in Section 6.5.1, the preamble results in a periodic waveform being received by the receiver electronics in each DTE which acts as a reference clock. The *start-of-frame delimiter* (*SFD*) is the single byte 10101011 which immediately follows the preamble and signals the start of a valid frame to the receiver.

(a)



SFD = start-of-frame delimiter
DA/SA = source/destination address
I/G = individuol (=0)/group (=1) address

FCS = frame check sequence
L/C = locally administered (=1)/
centrally administered (=0)

(b)

Bit rate	10 Mbps (Manchester encoded)
Slot time	512 bit times
Interframe gap	9.6 microseconds
Attempt limit	16
Back off limit	10
Jam size	32 bits
Maximum frame size (including FCS)	1518 bytes
Minimum frame size (including FCS)	512 bits

Figure 8.3 Ethernet/IEEE802.3 characteristics: (a) frame format; (b) operational parameters.

The *destination* and *source addresses* – also known as **MAC addresse** because they are used by the MAC sublayer – specify the identity of the har ware interface of both the intended destination station(s) and the originat station, respectively. Each address field can be either 16 or 48 bits, but for a particular LAN installation the size must be the same for all stations. The fi bit in the destination address field specifies whether the address is an **indiv** **ual address** (=0) or a **group address** (=1). If an individual address is speci the transmitted frame is intended for a single destination. If a group addr is specified, the frame is intended either for a logically related group of s tions (group address) or for all other stations connected to the netwo (**broadcast** or **global address**). In the latter case, the address field is set to binary 1s and, for a group address, the address specifies a previously agre group of stations. The type of grouping is specified in the second bit and be locally administered (=1) or centrally administered (=0). Group address are used for multicasting and the MAC unit/circuit associated with each s tion in the multicast group is then programmed to read all frames with th group address at its head.

With the original Ethernet standard, the two-byte *type* field immediately follows the address fields and indicates the network layer protocol that created the information in the data field. With the more recent IEEE802.3 format, the next two bytes are used as a *length indicator* which indicates the number of bytes in the data field. If this value is less than the minimum number required for a valid frame (minimum frame size), a sequence of bytes is added, known as **padding**. The maximum size of the data field – normally referred to as the **maximum transmission unit** (**MTU**) – is 1500 bytes. Finally, the *frame check sequence* (*FCS*) field contains a four-byte (32-bit) CRC value that is used for error detection. Note that with the original Ethernet standard, the end of a frame is detected when signal transitions end.

Frame transmission and reception

The frame transmission sequence is summarized in Figure 8.4(a). When a frame is to be transmitted, the frame contents are first encapsulated by the MAC unit into the format shown in Figure 8.3(a). To avoid contention with other transmissions on the medium, the MAC unit first monitors the carrier sense signal and, if necessary, defers to any passing frame. After a short additional delay (known as the **interframe gap**) to allow the passing frame to be received and processed by the addressed station(s), transmission of the frame is initiated.

As the bitstream is transmitted, the transmitter simultaneously monitors the received signal to detect whether a collision has occurred. Assuming a collision has not been detected, the complete frame is transmitted and, after the FCS field has been sent, the MAC unit awaits the arrival of a new frame, either from the cable or from the LC sublayer within the station. If a collision is detected, the transmitter immediately turns on the collision detect signal and enforces the collision by transmitting the jam sequence to ensure that the collision is detected by all other stations involved in the collision. After the jam sequence has been sent, the MAC unit terminates the transmission of the frame and schedules a retransmission attempt after a short randomly-computed interval.

Figure 8.4(b) summarizes the frame reception sequence. The MAC unit first detects the presence of an incoming signal and switches on the carrier sense signal to inhibit any new transmissions from this station. The incoming preamble is used to achieve bit synchronization and, when the start-of-frame delimiter has been detected, with an IEEE802.3 LAN, the length indicator is read and used to determine the number of bytes that follow. The frame contents including the destination and source addresses are then received and loaded into a frame buffer to await further processing. The received FCS field is first compared with the computed FCS and, if they are equal, the frame content is further checked to ensure it contains an integral number of bytes and that it is neither too short nor too long. If any of these checks fail then the frame is discarded. If all checks pass, then the destination address is read from the head of the frame and, if the frame is intended for this station – that is, the address of the station is the same as that in the frame or, if it is a group address, the station is a member of the specified group – the frame contents are passed to the LC sublayer for processing.

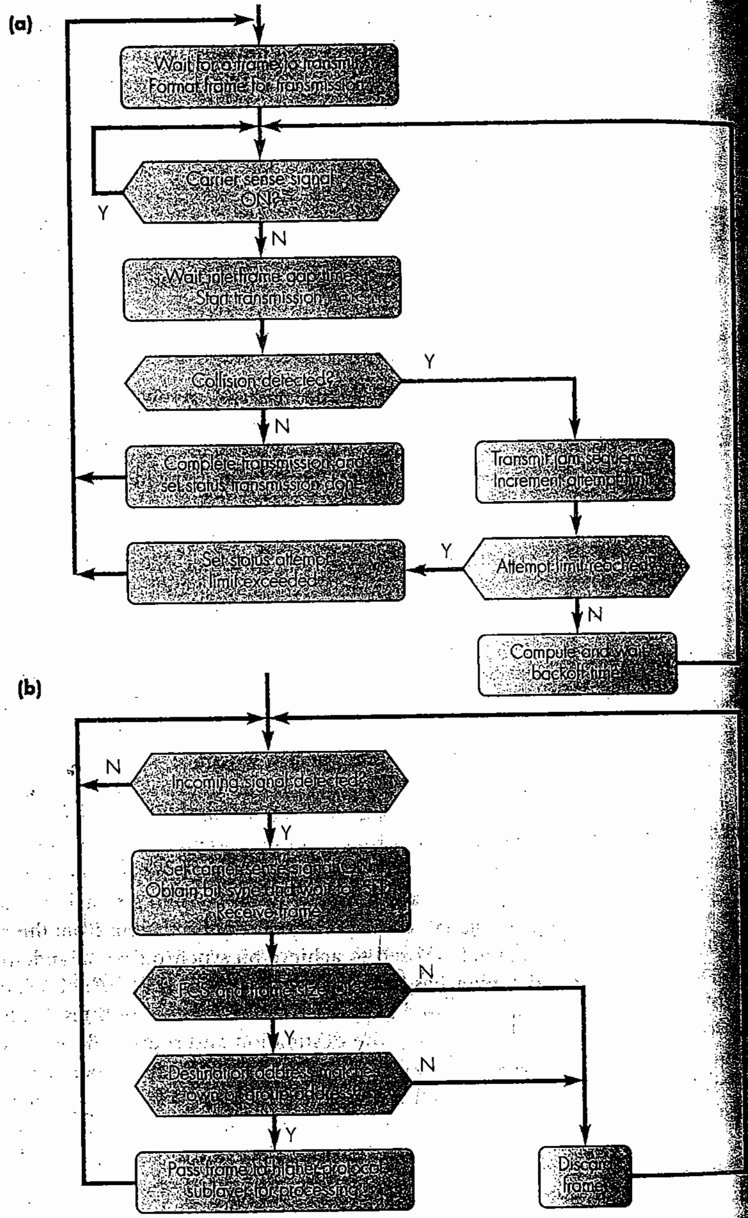

(a)
Wait for a frame to transmit
Format frame for transmission
Carrier sense signal ON?
Y
N
Wait interframe gap time
Start transmission
Collision detected?
Y
N
Complete transmission and set status transmission done
Transmit jam sequence
Increment attempt limit
Set status attempt limit exceeded
Y
Attempt limit reached?
N
Compute and wait backoff time
(b)
N
Incoming signal detected?
Y
Set carrier sense signal ON
Obtain bit sync and wait for SFD
Receive frame
FCS and frame size OK?
N
Y
Destination address matches own or group address?
N
Y
Pass frame to higher protocol sublayer for processing
Discard frame

Figure 8.4 CSMA/CD MAC sublayer operation: (a) transmit; (b) receive.

8.4 Token ring

Token ring networks are also used in technical and office environments and, in addition, in industrial environments. As we can deduce from the previous section, with an Ethernet LAN the time to transmit a frame is nondeterministic since, during heavy load conditions when collisions are likely to be frequent, the transmission of a frame may be delayed and, in the limit, unsuccessful. Although in an office environment this is to a degree acceptable, in industrial environments such as manufacturing and process control, it is unacceptable. Hence although token ring LANs also use a high bit rate shared/broadcast transmission medium, in order to provide a deterministic service, they utilize a completely different MAC method.

Also, because the type of information – and hence frame contents – to be transmitted in an industrial application may have different levels of importance, frames can be assigned different priorities. The MAC method, therefore, also contains a priority control algorithm to ensure higher priority frames – for example those containing alarm messages – are transmitted before lower priority frames. We shall discuss this and other issues separately.

Control token

In a token ring LAN, all the stations are connected together by a set of unidirectional links in the form of a ring and all frame transmissions between any of the stations take place over it by circulating the frame around the ring. In its basic form, only one frame transfer can be in progress over the ring at a time. When the ring is first initialized, a single control token (frame) is generated and, in the absence of any frames to transmit, this continuously circulates around the ring. Then, when a station generates a frame to send, the steps taken by the MAC unit within each station are as illustrated in Figure 8.5.

Whenever a station wishes to send a frame, it first waits for the token. On receipt of the token, it initiates transmission of the frame, which includes the address of the intended recipient at its head. The frame is repeated (that is, each bit is received and then retransmitted) by all the stations in the ring until it circulates back to the initiating station, where it is removed. In addition to repeating the frame, each station reads and stores the frame contents. The intended recipient – indicated by the destination address in the frame header – retains a copy of the frame and indicates that it has done this by setting the response bits at the tail of the frame.

A station releases the token in one of two ways depending on the bit rate (speed) of the ring. With slower rings (4 Mbps), the token is released only after the response bits have been received. With higher speed rings (16 Mbps), it is released immediately after transmitting the last bit of the frame. This is known as **early (token) release** and, as we shall explain later, this is done to improve the level of utilization of the ring.

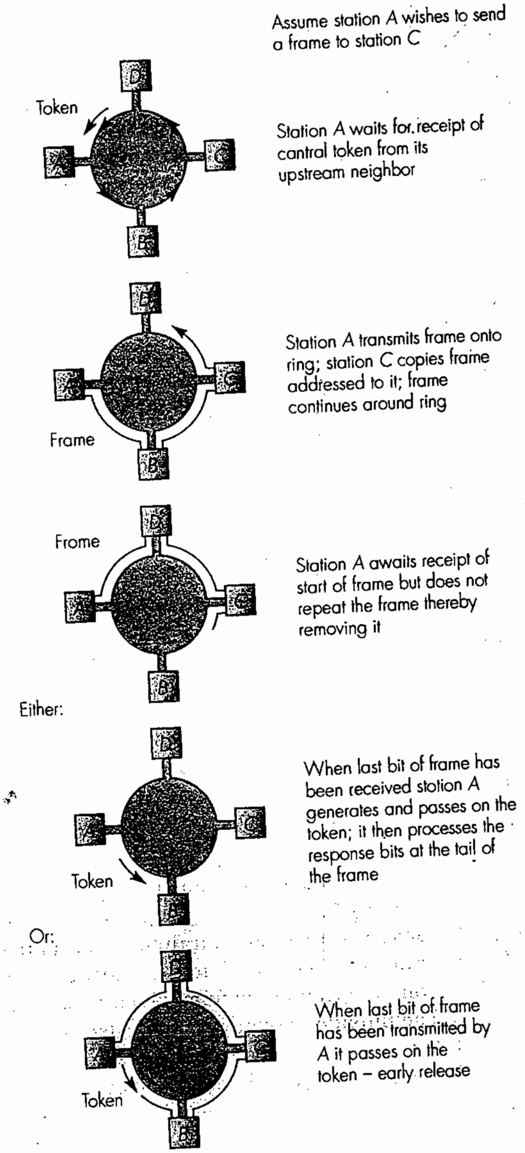

Assume station A wishes to send a frame to station C
Token
D
A
C
B
Station A waits for receipt of control token from its upstream neighbor
D
A
C
B
Frame
Station A transmits frame onto ring; station C copies frame addressed to it; frame continues around ring
Frame
D
A
C
B
Station A awaits receipt of start of frame but does not repeat the frame thereby removing it
Either:
D
A
C
B
Token
When last bit of frame has been received station A generates and passes on the token; it then processes the response bits at the tail of the frame
Or:
D
A
C
B
Token
When last bit of frame has been transmitted by A it passes on the token – early release

Figure 8.5 Token ring network: principle of operation.

Wiring configurations

A small token ring network is shown in Figure 8.6(a). As we can see, in this configuration is a single hub – also known as a **concentrator** and located in, say, the wiring closet of an office – to which all the stations are attached by a cable containing two twisted-pairs, one for each direction of transmission. Within the hub, associated with each station is a **station coupling unit (SCU)** and, as we show in the figure, these are interconnected so that all the stations are interconnected together to form a unidirectional ring.

An SCU is shown in Figure 8.6(b) and, as we can see, it contains a set of relays and additional electronics to drive and receive signals to and from the cable. The relays are so arranged that whenever a station is switched off, the SCU is in the *bypassed state* and a continuous transmission path through the SCU is maintained. The insertion of a station into the ring is initiated when the station is switched on. A separate pair of wires in the drop cable is used to pass power from the station to the SCU and, when activated, the relays change position so that the station becomes inserted.

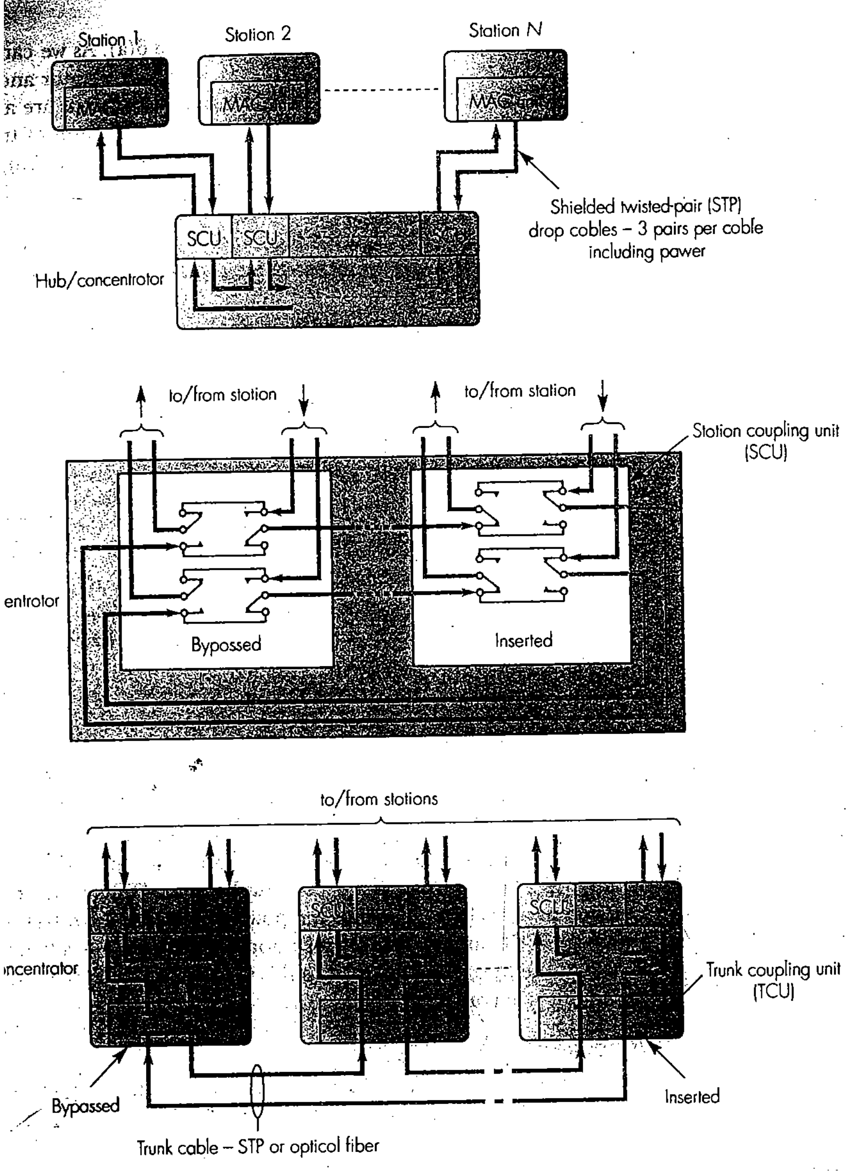
When the SCU is in the *inserted state*, all signals from the ring are routed through the MAC unit of the station. The receive/transmit electronics in the MAC unit then either simply read and relay (repeat) the received signal to the transmit side, if this station is not the originator of the frame, or remove the received signal from the ring, if it initiated the transmission.

The use of two pairs of relays connected in this way means that the MAC unit can detect certain open-circuit and short-circuit faults in either the transmit or the receive pair of signal wires. Also, in the bypassed state, the MAC unit can conduct self-test functions, since any data output on the transmit pair is looped back on the receive pair. Each station is connected to the SCU by a shielded twisted-pair (STP) cable containing three twisted-pair wires; one for transmission, the second for reception, and the third to supply power to the SCU.

As we show in Figure 8.6(c), larger configurations are formed by interconnecting multiple nodes/concentrators together by means of either an STP or optical fiber trunk cable. In this case, associated with each hub is a second relay unit known as a **trunk coupling unit (TCU)**. This has the same function as an SCU except that the power to activate the relays – and hence insert the TCU and its attached stations into the ring – is from the power supply of the hub/concentrator. In this way, should a hub fail, the remainder of the ring will continue functioning.

Ring interface

The MAC unit in each station performs such functions as frame encapsulation and de-encapsulation, FCS generation and error detection, and the implementation of the MAC control algorithm. Associated with the ring is a single master station known as the **active ring monitor** that supplies the master clock for the ring. All stations are capable of performing this function and the active monitor is selected using a bidding process involving all the stations that are currently inserted into the ring.


Station 1
Station 2
Station N
MAC unit
SCU
SCU
Shielded twisted-pair (STP) drop cables – 3 pairs per cable including power
Hub/concentrator
to/from station
to/from station
Station coupling unit (SCU)
entrator
Bypassed
Inserted
to/from stations
ncentrator
SCU
SCU
Trunk coupling unit (TCU)
Bypassed
Inserted
Trunk cable – STP or optical fiber

Token ring wiring configurations: (a) single hub; (b) station coupling unit; le hubs/concentrators.

Each circulating bitstream is differential Manchester encoded by the active ring monitor and all other inserted stations in the ring frequency and phase lock their own clock using a DPLL and the incoming bitstream as we explained in Section 6.5.1. Hence there is only a small fraction of a bit delay in each ring interface as the repeater retransmits each bit as it is being received. In addition, when the station is the active ring monitor, it ensures the ring has a **minimum latency time**. This is the time, measured in bit times at the ring data transmission rate, taken for a signal to propagate once around the ring. The ring latency time includes the signal propagation delay through the ring transmission medium together with the sum of the delays through each MAC unit. For the control token to circulate continuously around the ring when none of the stations requires to use the ring (that is, all stations are simply in the repeat mode), the ring must have a minimum latency time of at least the number of bits in the token sequence to ensure that the token is not corrupted.

The token is 24 bits long, so when a station is the active ring monitor, its MAC unit provides a fixed 24-bit buffer, which effectively becomes part of the ring to ensure its correct operation under all conditions. Although the mean line signaling rate around the ring is controlled by a single master clock in the active monitor, the use of a separate DPLL circuit in each MAC unit means that the actual signaling rate may vary slightly around the ring. The worst-case variation is when the maximum number of stations (250) are all active, which is equivalent to plus or minus three bits. Unless the latency of the ring remains constant, however, bits will be corrupted as the latency decreases, or additional bits will be added as the latency increases. To maintain a constant ring latency, an additional **elastic (variable) buffer** with a length of six bits is added to the fixed 24-bit buffer. The resulting 30-bit buffer is initialized to 27 bits. If the received signal at the master MAC unit is faster than the master oscillator, the buffer is expanded by a single bit. Alternatively, if the received signal is slower, the buffer is reduced by a single bit. In this way the ring always comprises sufficient bits to allow the token to circulate continuously around the ring in the quiescent (idle) state.

Frame formats

In addition to the control token and information frames, additional frames are used for various ring management functions including the selection of an active ring monitor. Collectively, these are known as MAC frames.

Two basic formats are used in token rings: one for the control token and the other for normal frames. The control token is the means by which the right to transmit (as opposed to the normal process of repeating) is passed from one station to another, whereas a normal frame is used by a station to send either data or MAC information around the ring. The format of the two types of frame are given in Figure 8.7 together with the bit sequence used for each field.

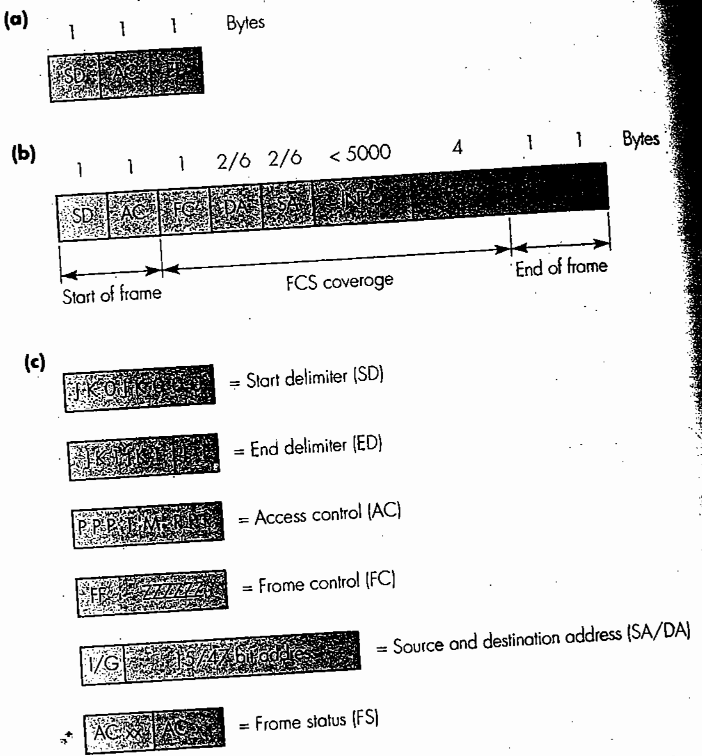

(a)
1 1 1 Bytes
SD AC ED
(b)
1 1 1 2/6 2/6 < 5000 4 1 1 Bytes
SD AC FC DA SA INFO
Start of frame
FCS coverage
End of frame
(c)
J K 0 J K 0 0 0 = Start delimiter (SD)
J K 1 J K 1 I E = End delimiter (ED)
P P P T M R R R = Access control (AC)
F F Z Z Z Z Z Z = Frame control (FC)
I/G 15/47 bit address = Source and destination address (SA/DA)
A C x x A C x x = Frame status (FS)

Figure 8.7 Token ring network frame formats and field descriptions: (a) token format; (b) frame format; (c) field descriptions.

The *start delimiter* (*SD*) and *end delimiter* (*ED*) fields are specia
sequences used to achieve data transparency. They exploit the sym
encoding method used on the cable medium: all information bits trans
ted on the medium are differential Manchester encoded, except for selec
bits in the SD and ED fields. In contrast, the J and K symbols depart from
normal encoding rules, being used instead to represent constant levels
the complete bit cell period. The J symbol has the same polarity as
preceding symbol, whereas a K symbol has the opposite polarity to
preceding symbol. In this way the receiver can reliably detect the sta
and end of each transmitted token or frame irrespective of its contents
length. Note, however, that only the first six symbols (JK1JK1 in Figu
8.7(c)) are used to indicate a valid end of frame. The other two bits, I and
have other functions:

- In a token, both the I- and E-bits are 0.
- In a normal frame, the I-bit is used to indicate whether the frame is the first (or an intermediate) frame in a sequence (I = 1) or the last (or only) frame (I = 0).
- The E-bit is used for error detection. It is set to 0 by the originating station but, if any station detects an error while receiving or repeating the frame (for example, FCS error), it sets the E-bit to 1 to signal to the originating station that an error has been detected.

The *access control* (*AC*) field comprises the priority bits, the token and monitor bits, and the reservation bits. As its name implies, the AC field is used to control access to the ring. When it is part of the token, the priority bits (P) indicate the priority of the token and hence which frames a station may transmit on receipt of the token. The token bit (T) discriminates between a token and an ordinary frame (0 indicates a token, 1 a frame). The monitor bit (M) is used by the active monitor to prevent a frame from circulating around the ring continuously. Finally, the reservation bits (R) allow stations holding high-priority frames to request (in either repeated frames or tokens) that the next token to be issued is of the requisite priority.

The *frame control* (*FC*) field defines the type of the frame (MAC or information) and certain control functions. If the frame type bits (F) indicate a MAC frame, all stations on the ring interpret and, if necessary, act on the control bits (Z). If it is an I-frame, the control bits are interpreted only by the stations identified in the destination address field. Both the *source* and *destination addresses* (*SA* and *DA*) are standard 16/48 bit MAC addresses.

The *information* (*INFO*) field is used to carry either user data or additional control information when included in a MAC frame. Although no maximum length is specified for the information field, it is limited in practice by the maximum time which a station is allowed to transmit a frame when holding the control token. A typical maximum length is 5000 bytes.

The *frame check sequence* (*FCS*) field is derived from a 32-bit CRC. Finally, the *frame status* (*FS*) field is made up of two fields: the address-recognized bits (A) and the frame-copied bits (C). Both the A- and C-bits are set to 0 by the station originating the frame. If the frame is recognized by one or more stations on the ring, the station(s) sets the A-bits to 1. Also, if it copies the frame, it sets the C-bits to 1. In this way, the originating station can determine whether the addressed station(s) is non-existent or switched off, is active but did not copy the frame, or is active and copied the frame.

Frame transmission

On receipt of a service request to transmit a block of data (which includes the destination address and the priority of the data as a parameter), the data is first encapsulated by the MAC unit into the standard format shown in Figure 8.7. The MAC unit awaits the reception of a token with a priority less than or equal to the priority of the assembled frame. Clearly, in a system that employs multiple priorities, a procedure must be followed to ensure that all stations have an opportunity to transmit frames in the correct order of priority. This procedure works as follows.

After formatting a frame and prior to receiving an appropriate token (that is, one with a priority less than or equal to the priority of the waiting frame), each time a frame or a token with a higher priority is repeated at the ring interface, the MAC unit reads the value of the reservation bits contained within the AC field. If these are equal to or higher than the priority of the waiting frame, the reservation bits are simply repeated unchanged. If they are lower, the MAC unit replaces the current value with the priority of the waiting frame. Then, assuming there are no other higher priority frames awaiting transmission on the ring, the token is passed on by the current owner (user) with this priority. On receipt of the token, the waiting MAC unit detects that the priority of the token is equal to the priority of the frame it has waiting to be transmitted. It therefore accepts the token by changing the token bit in the AC field to 1, prior to repeating this bit, which effectively converts the token to a start-of-frame sequence for a normal frame. The MAC unit then stops repeating the incoming signal and follows the converted start-of-frame sequence with the preformatted frame contents. While the frame contents are being transmitted, the FCS is computed and subsequently appended after the frame contents, before transmitting the end-of-frame sequence.

Once transmission of the waiting frame(s) has been started, the MAC unit stops repeating, thus removing the transmitted frame(s) after it has circulated the ring. In addition, the MAC unit notes the state of the A- and C-bits in the FS field at the tail of the frame(s) to determine whether the frame(s) has (have) been copied or ignored. It then generates a new token and forwards this on the ring to allow another waiting station to gain access to the ring. More than one frame may be sent on receipt of a usable token providing, firstly, that the priority of the other waiting frame(s) is greater than or equal to the priority of the token and, secondly, that the total time taken to transmit the other frame(s) is within a defined limit known as the **token holding time**. The default setting for the latter is 10 ms. The frame transmission operation is illustrated in Figure 8.8(a).

Frame reception

In addition to repeating the incoming signal/bitstream, the MAC unit within each active station on the ring detects the start of each frame by recognizing the special start-of-frame bit sequence. It then determines whether the frame should simply be repeated or copied. If the F-bits indicate that it is a MAC frame, the frame is copied and the C-bits are interpreted and, if necessary, acted upon. However, if the frame is a normal data-carrying frame and the DA matches either the station's individual address or relevant group address, the frame contents are copied into a frame buffer and passed on to the LC sublayer for further processing. In either case, the A- and C-bits in the frame status field at the tail of the frame are set accordingly prior to being repeated. The reception operation is shown in Figure 8.8(b).

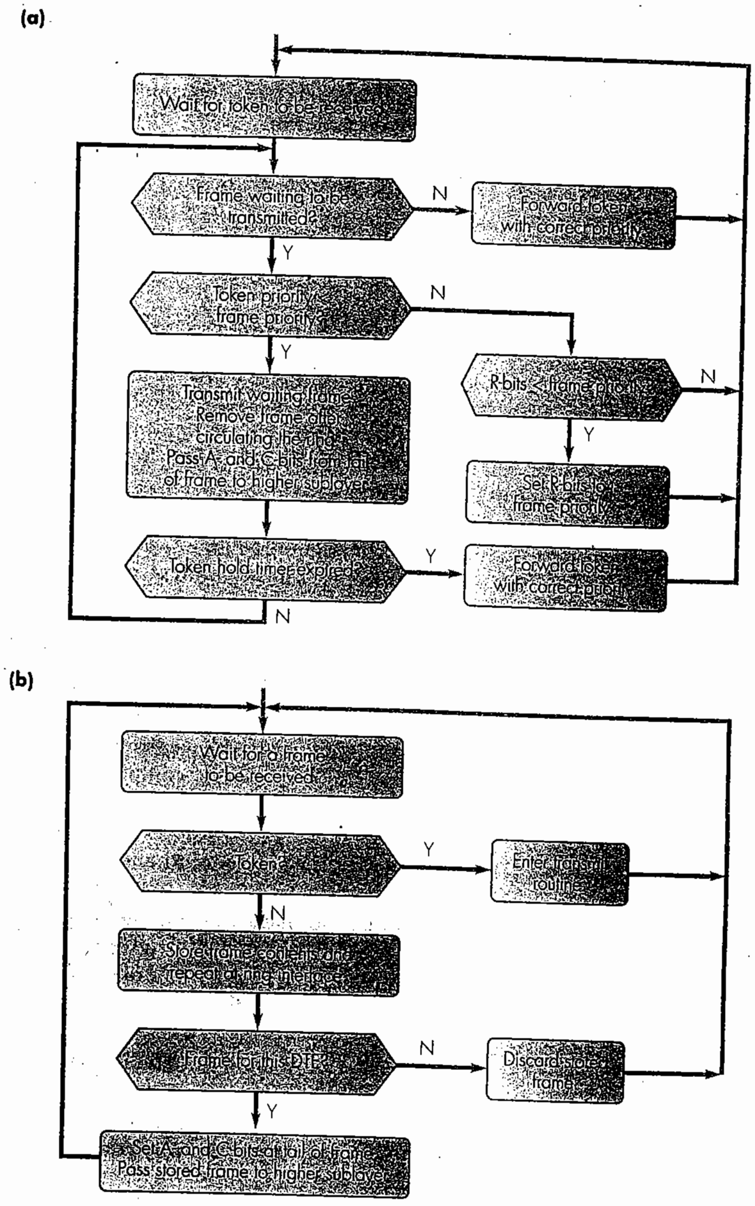

(a)
Wait for token to be received
Frame waiting to be transmitted?
N
Forward token with correct priority
Y
Token priority ≤ frame priority?
N
Y
R-bits < frame priority?
N
Y
Transmit waiting frame
Remove frame after circulating the ring
Pass A- and C-bits from tail of frame to higher sublayer
Set R-bits to frame priority
Token hold timer expired?
Y
Forward token with correct priority
N
(b)
Wait for a frame to be received
Token?
Y
Enter transmit routine
N
Store frame contents and repeat at ring interface
Frame for this DTE?
N
Discard stored frame
Y
Set A- and C-bits at tail of frame
Pass stored frame to higher sublayer

Figure 8.8 Token ring MAC sublayer operation: (a) transmit; (b) receive.

Priority operation

The priority assigned to a token by a MAC unit after it has completed tra mitting any waiting frame(s) is determined by a mechanism that endeavors ensure both of the following:

(1) Frames with a higher priority than the current ring service priority are always transmitted on the ring first.

(2) All stations holding frames with the same priority have equal access righ to the ring.

This is accomplished by using both the P- and the R-bits in the AC field o each frame coupled with a mechanism that ensures a station that raises the service priority level of the ring returns the ring to its original level after the higher priority frames have been transmitted.

To implement this scheme, each MAC unit maintains two sets of values The first set comprises three variables Pm, Pr, and Rr. Pm specifies the highest priority value contained within any of the frames currently waiting transmission at the station. Pr and Rr are known as **priority registers** and contain, respectively, the priority and reservation values held within the AC field of the most recently repeated token or frame. The second set of values comprises two stacks known as the Sr and Sx stacks which are used as follows.

All frames transmitted by a station, on receiving a usable token, are assigned a priority value in the AC field equal to the present ring service pr ority Pr, and a reservation value of zero. After all waiting frames at or greate than the current ring priority have been transmitted, or until the transmission of another frame cannot be completed before the token holding tim expires, the MAC unit generates a new token with:

(1) P = Pr and R = the greater of Rr and Pm

if the station does not have any more waiting frames with a priority (as co tained in register Pm) equal to or greater than the current ring servi priority (as contained in register Pr), or does not have a reservation reques (as contained in register Rr) greater than the current priority.

(2) P = the greater of Rr and Pm and R = 0

if the station has another waiting frame(s) with a priority (as contained i Pm) greater than the current priority Pr, or if the current contents of Rr ar greater than the current priority.

Since in the latter case the station effectively raises the service priority level of the ring, it becomes what is known as a **stacking station** and, as such, stores the value of the old ring service priority (Pr) on stack Sr and the new

ring service priority (P) on stack Sx. These values are saved, as it is the responsibility of the station that becomes the stacking station to lower the ring service priority level when there are no frames ready to transmit, at any point on the ring, with a priority equal to or greater than the P stacked on Sx. Also, a stack is used rather than a single register because a stacking station may need to raise the service priority of the ring more than once before the service priority is returned to a lower priority level. The different values assigned to the P- and R-bits of the token and the actions performed on the two stacks are summarized in Figure 8.9(a).

Having become a stacking station, the MAC unit claims every token that it receives with a priority equal to that stacked on Sx to examine the

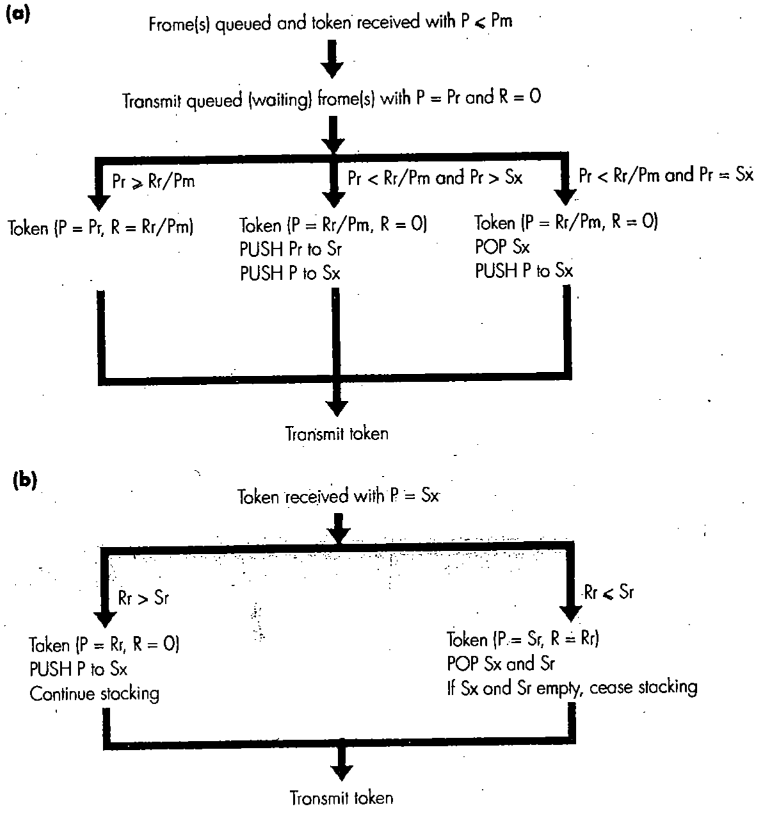

(a)
Frame(s) queued and token received with P ≤ Pm
Transmit queued (waiting) frame(s) with P = Pr and R = 0
Pr ≥ Rr/Pm
Pr < Rr/Pm and Pr > Sx
Pr < Rr/Pm and Pr = Sx
Token (P = Pr, R = Rr/Pm)
Token (P = Rr/Pm, R = 0)
PUSH Pr to Sr
PUSH P to Sx
Token (P = Rr/Pm, R = 0)
POP Sx
PUSH P to Sx
Transmit token
(b)
Token received with P = Sx
Rr > Sr
Rr ≤ Sr
Token (P = Rr, R = 0)
PUSH P to Sx
Continue stacking
Token (P = Sr, R = Rr)
POP Sx and Sr
If Sx and Sr empty, cease stacking
Transmit token

Figure 8.9 Token generation and stack modifications: (a) token generation [Note: Sx = 0 if stack empty]; (b) stack modification.

value in the R-bits of the AC field to determine if the service priority of the ring should be raised, maintained, or lowered. The new token is then transmitted with:

(1) P = Rr and R = 0

if the value contained in the R-bits (the current contents of register Rr) is greater than Sr. The new ring service priority (P) is stacked (PUSHed) on to Sx and the station continues its role as a stacking station.

(2) P = Sr and R = Rr (unchanged)

if the value contained in the R-bits is less than or equal to Sr. Both values currently on the top of stacks Sx and Sr are POPped from the stack and, if both stacks are then empty, the station discontinues its role as a stacking station. These two operations are summarized in Figure 8.9(b).

mple 8.1

A token ring network has been configured to operate with four priority classes: 0, 2, 4, and 8 with 8 the highest priority. After a period of inactivity when no transmissions occur on successive rotations of the token, four stations generate frames to send as follows:

Station 1 : 1 frame of priority 2
Station 7 : 1 frame of priority 2
Station 15 : 1 frame of priority 4
Station 17 : 1 frame of priority 4

Assuming the order of stations on the ring is in increasing numerical order and that station 1 receives the token first with a zero priority and reservation field, derive and show in table form the transmissions made by each station for the next eight rotations of the token. Include in your table the values in the priority and reservation fields both as each new token is generated and as each frame circulates around the ring. Also include the actions taken by the stacking station.

Answer:

The transmissions made by each station for the next eight rotations of the token are shown in Figure 8.10.

On the first rotation of the token, station 1 seizes the token and initiates the transmission of its waiting frame. Also on this rotation the reservation field in the frame is raised by station 7 to 2 and then by station 15 to 4.

8.1 Continued

On the second rotation, station 1 reads the reservation field from the frame and determines it must release the token with a priority of 4. Since it is raising the ring priority, it must become a stacking station and saves the current ring priority (0) on stack Sr and the new priority (4) on Sx. The token then rotates and is seized by station 15. Also on this rotation station 17 raises the reservation field from 0 to 4.

On the third rotation, station 15 releases the token with a priority and reservation field of 4. Station 17 therefore seizes the token and initiates the transmission of its waiting frame.

On the fourth rotation, station 7 updates the reservation field from 0 to 2 and this causes the token to be released by station 17 with the same priority (4) but a reservation value of 2.

On the fifth rotation, since station 1 is a stacking station, it detects Rr is greater than Sr and hence lowers the priority of the token/ring from 4 to 2 and saves the lower priority on the stack. Station 7 is therefore able to transmit its waiting frame.

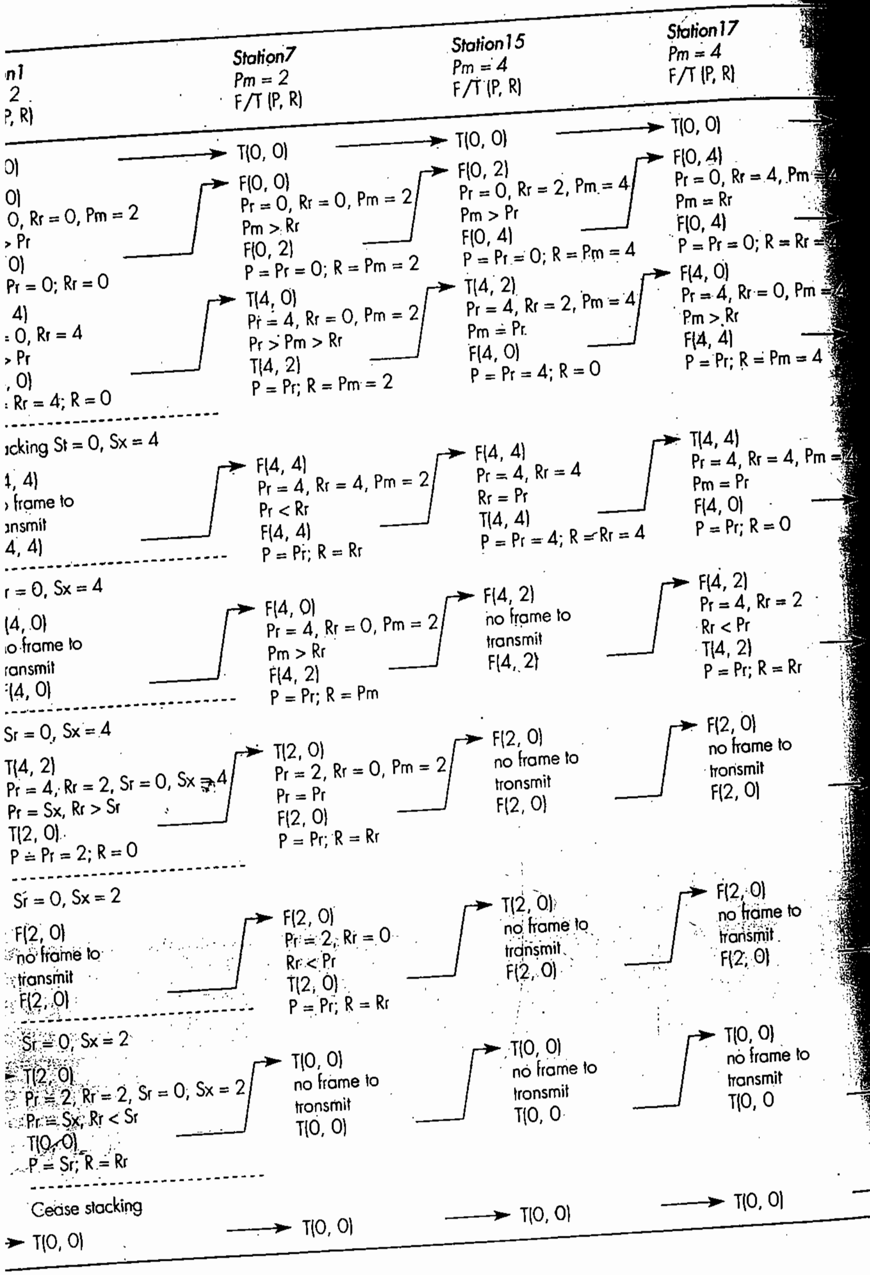
On the sixth rotation, station 7 releases the token with the same priority since no reservations have been made.

On the seventh rotation, station 1 detects the reservation field in the token is less than the priority field and hence reduces the priority to 0 and thereby ceases to be a stacking station. The token has thus returned to its initial state and continues rotating until further frames are generated.

Ring management

We have been primarily concerned with the transmission of frames and tokens during normal operation of the ring. However, the ring must be set up before normal operation can take place. If a station wishes to join an already operational ring, the station must first go through an initialization procedure to ensure that it does not interfere with the correct functioning of the established ring. In addition, during normal operation it is necessary for each active station on the ring to monitor continuously its correct operation and, if a fault develops, to take corrective action to try to re-establish a correctly functioning ring. Collectively, these functions are known as **ring management.** A list of the various MAC frame types associated with these functions is given in Table 8.1 and a summary of the role of each function follows.

Initialization: When a station wishes to become part of the ring after being either switched on or reset, it enters an initialization sequence to ensure that no other stations in the ring are using the same address and to inform its immediate downstream neighbor that it has (re)entered the ring.


n1
2
P, R)
Station7
Pm = 2
F/T (P, R)
Station15
Pm = 4
F/T (P, R)
Station17
Pm = 4
F/T (P, R)
0)
0)
0, Rr = 0, Pm = 2
> Pr
0)
Pr = 0; Rr = 0
4)
= 0, Rr = 4
> Pr
, 0)
Rr = 4; R = 0
T(0, 0)
F(0, 0)
Pr = 0, Rr = 0, Pm = 2
Pm > Rr
F(0, 2)
P = Pr = 0; R = Pm = 2
T(4, 0)
Pr = 4, Rr = 0, Pm = 2
Pr > Pm > Rr
T(4, 2)
P = Pr; R = Pm = 2
T(0, 0)
F(0, 2)
Pr = 0, Rr = 2, Pm = 4
Pm > Pr
F(0, 4)
P = Pr = 0; R = Pm = 4
T(4, 2)
Pr = 4, Rr = 2, Pm = 4
Pm = Pr
F(4, 0)
P = Pr = 4; R = 0
T(0, 0)
F(0, 4)
Pr = 0, Rr = 4, Pm =
Pm = Rr
F(0, 4)
P = Pr = 0; R = Rr =
F(4, 0)
Pr = 4, Rr = 0, Pm =
Pm > Rr
F(4, 4)
P = Pr; R = Pm = 4
acking St = 0, Sx = 4
4, 4)
frame to
ansmit
4, 4)
F(4, 4)
Pr = 4, Rr = 4, Pm = 2
Pr < Rr
F(4, 4)
P = Pr; R = Rr
F(4, 4)
Pr = 4, Rr = 4
Rr = Pr
T(4, 4)
P = Pr = 4; R = Rr = 4
T(4, 4)
Pr = 4, Rr = 4, Pm =
Pm = Pr
F(4, 0)
P = Pr; R = 0
r = 0, Sx = 4
(4, 0)
o frame to
ransmit
(4, 0)
F(4, 0)
Pr = 4, Rr = 0, Pm = 2
Pm > Rr
F(4, 2)
P = Pr; R = Pm
F(4, 2)
no frame to
transmit
F(4, 2)
F(4, 2)
Pr = 4, Rr = 2
Rr < Pr
T(4, 2)
P = Pr; R = Rr
Sr = 0, Sx = 4
T(4, 2)
Pr = 4, Rr = 2, Sr = 0, Sx = 4
Pr = Sx, Rr > Sr
T(2, 0)
P = Pr = 2; R = 0
T(2, 0)
Pr = 2, Rr = 0, Pm = 2
Pr = Pr
F(2, 0)
P = Pr; R = Rr
F(2, 0)
no frame to
transmit
F(2, 0)
F(2, 0)
no frame to
transmit
F(2, 0)
Sr = 0, Sx = 2
F(2, 0)
no frame to
transmit
F(2, 0)
F(2, 0)
Pr = 2, Rr = 0
Rr < Pr
T(2, 0)
P = Pr; R = Rr
T(2, 0)
no frame to
transmit
F(2, 0)
F(2, 0)
no frame to
transmit
F(2, 0)
Sr = 0, Sx = 2
T(2, 0)
Pr = 2, Rr = 2, Sr = 0, Sx = 2
Pr = Sx, Rr < Sr
T(0, 0)
P = Sr; R = Rr
T(0, 0)
no frame to
transmit
T(0, 0)
T(0, 0)
no frame to
transmit
T(0, 0
T(0, 0)
no frame to
transmit
T(0, 0
Cease stacking
T(0, 0)
T(0, 0)
T(0, 0)
T(0, 0)

8.10 Token ring priority example.

Table 8.1 Token ring management: MAC frame types.

Frame type	*Function*
Duplicate address test (DAT)	Used during the initialization procedure to enable a station to determine that no other stations in the ring are using its own address
Standby monitor present (SMP)	Used in the initialization procedure to enable a station to determine the address of its upstream neighbor (successor) in the ring
Active monitor present (AMP)	These types of frames are transmitted at regular intervals by the currently active monitor and each station monitors their passage
Claim token (CT)	Used in the procedure to determine a new active monitor if the current one fails
Purge (PRG)	Used by a new active monitor to initialize all stations into the idle state
Beacon (BCN)	Used in the beaconing procedure

Standby monitor: Upon completion of the initialization sequence, the station can start to transmit and receive normal frames and tokens. In addition, the station enters the standby monitor state to monitor continuously the correct operation of the ring. It does this by monitoring the passage of tokens and special active monitor present (AMP) MAC frames – which are periodically transmitted by the currently active monitor – as they are repeated at the ring interface.

Active monitor: If the station is successful in its bid to become the new active monitor, it first inserts its latency buffer into the ring and enables its own clock. It then initiates the transmission of a purge (PRG) MAC frame to ensure that there are no other tokens or frames on the ring before it initiates the transmission of a new token.

Beaconing: If a serious failure such as a broken cable arises in the ring, a procedure known as beaconing informs each station on the ring that the token-passing protocol has been suspended (until the affected failure domain has been located and repaired). The failure domain consists of the following:

- the station that reports the failure, which is referred to as the **beaconing station;**
- the station upstream of the beaconing station;
- the ring medium between them.

We can see that the MAC procedures used with a token ring network are quite complicated, certainly compared with a CSMA/CD bus, for example. Remember, however, that most of the procedures are implemented in special integrated circuits within the MAC unit, so their operation is transparent to the user. Moreover, many of these ring management procedures are invoked only when faults develop and so the overheads associated with them are, on the whole, modest.

8.5 Bridges

There are two types of bridge, the ones that are used with Ethernet LANs, known as **transparent bridges**, and the others with token ring **LANs**, known as **source routing bridges**. Before we explain their operation, however, it will be helpful first to review the operation of repeaters since bridges were designed to overcome the limitations that occur when using repeaters.

Repeaters are used to ensure that the electrical signal transmitted by the line drivers within the MAC unit propagate throughout the network. For each LAN segment, in order to limit the signal attenuation to an acceptable level, there is a defined maximum limit set on the physical length of the segment and on the number of (end) stations that may be attached to it. When interconnecting segments, a repeater is used to limit the electrical drive requirements of the line driver circuit to that of a single segment. In this way the presence of multiple segments (and hence repeaters) in a transmission path is transparent to the source station. The repeater, after achieving clock synchronization, simply regenerates all signals received on one segment and forwards (repeats) them onto the next segment. This form of interconnection is shown in Figure 8.11(a) and, as we can deduce from the figure, all frame transmissions from any station attached to a segment will propagate throughout the total LAN and hence be received by all the other attached stations. This means, therefore, that in terms of available bandwidth, the network behaves like a single segment.

In early LAN installations, because most traffic was text-based email and occasional file transfers, this mode of operation gave an acceptable performance in terms of network access and transfer delays. With the arrival of diskless nodes/stations, however, all disk accesses to the server are via the network and hence the demands on the network bandwidth are substantially higher. In most cases, the server and the set of diskless nodes it serves are all attached to the same LAN segment. Hence there is no necessity for the frames associated with such transfers to be transmitted beyond the segment on which they are generated. Bridges were introduced, therefore, to inhibit the forwarding of such frame transfers and only to forward those frames that are intended for a different segment.

Thus, the function of a bridge is similar to a repeater in that it is used for interconnecting LAN segments. However, when bridges are used, all frames received from a segment are buffered (stored) and error checked before they are repeated (forwarded). Moreover, only frames that are free of errors and are addressed to stations on a different segment from the one on which they were received are forwarded. Consequently, all transmissions between stations connected to the same LAN segment are not forwarded and hence do not load the rest of the network. A bridge thus operates at the MAC sublayer in the context of our reference model. This is shown in Figure 8.11(b) and the resulting LAN is then referred to as a **bridged LAN**.

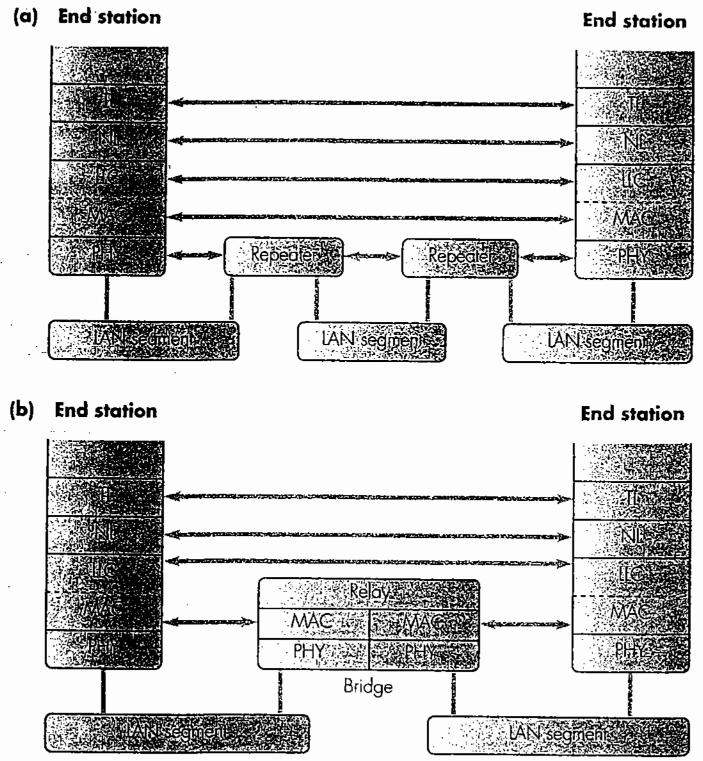


Figure 8.11 LAN interconnection: (a) repeaters; (b) bridges.

8.5.1 Transparent bridges

With a transparent bridge, as with a repeater, the presence of one (or more) bridges in a route between two communicating stations is transparent to the two stations. All routing decisions are made exclusively by the bridge(s). Moreover, a transparent bridge automatically initializes and configures itself

(in terms of its routing information) in a dynamic way after it has bee
into service. A schematic of a bridge is shown in Figure 8.12(a) and a s
bridged LAN in Figure 8.12(b).

A LAN segment is physically connected to a bridge through a b
port. A basic bridge has just two ports whereas a **multiport bridge**
number of connected ports (and hence segments). In practice, each b
port comprises the MAC integrated circuit chipset associated with the p
ular type of LAN segment – CSMA/CD, Ethernet – together with s
associated port management software. The software is responsible for in
izing the chipset at start-up – chipsets are all programmable devices –
for buffer management. Normally, the available memory is logically divi
into a number of fixed-size units known as buffers. Buffer managem
involves passing a free buffer (pointer) to the chipset for onward transm
sion (forwarding).

Every bridge operates in the **promiscuous mode** which means it rece
and buffers all frames received on each of its ports. When a frame has be
received at a port and put into the assigned buffer by the MAC chipset,
port management software prepares the chipset for a new frame and th
passes the pointer of the memory buffer containing the received frame to
bridge protocol entity for processing. Since two (or more) frames may arr
concurrently at the ports and two or more frames may need to be forwarde
from the same output port, the passing of memory pointers between the p
management software and the bridge protocol entity software is carried
via a set of queues.

As we shall describe later, each port may be in a number of alterna
states and processing of received frames is carried out according to a defin
protocol. The function of the bridge protocol entity software is to impleme
the particular bridge protocol being used.

Frame forwarding (filtering)

A bridge maintains a **forwarding database** (also known as a **routing directo**
that indicates, for each port, the outgoing port (if any) to be used for forwa
ing each frame received at that port. If a frame is received at a port th
addressed to a station on the segment (and hence port) on which it
received, the frame is discarded; otherwise it is forwarded via the port speci
in the forwarding database. The normal routing decision involves a sim
look-up operation: the destination address in each received frame is first re
and then used to access the corresponding port number from the forwardi
database. If this is the same as the port on which it was received, the frame
discarded, else it is queued for forward transmission on the segment associat
with the accessed port. This process is also known as **frame filtering.**

Bridge learning

A problem with transparent bridges is the creation of the forwarding dat
base. One approach is for the contents of the forwarding database to b
created in advance and held in a fixed memory, such as programmabl

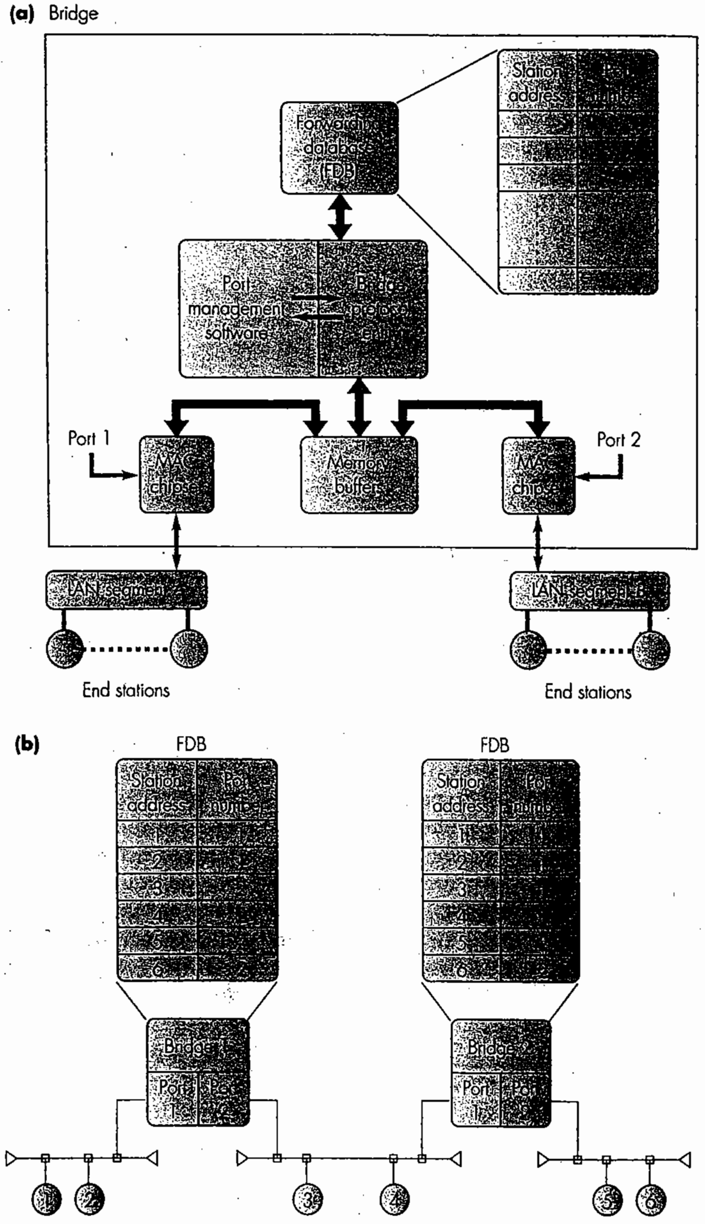


Figure 8.12 Transparent bridge schematic: (a) architecture; (b) application example.

read-only memory (PROM). The disadvantage is that the contents of the forwarding database in all bridges have to be changed whenever the network topology is changed – a new segment added, for example – or when a user changed the point of attachment (and hence segment) of his or her station. To avoid this, in most bridged LANs the contents of the forwarding database are not statically set up but rather are dynamically created and maintained during normal operation of the bridge. This is accomplished using a combination of a learning process and a dialog with other bridges to ascertain the topology of the overall installed LAN. An overview of the learning process is as follows.

When a bridge first comes into service, its forwarding database is initialized to empty. Whenever a frame is received, the *source address* within it is read and the incoming port number on which the frame was received is entered into the forwarding database. In addition, since the forwarding port is not known at this time, a copy of the frame is forwarded on all the other output ports of the bridge. As these frames propagate through the network, this procedure is repeated by each bridge. Firstly, the incoming port number is entered in the forwarding database against the source (station) address and a copy of the frame is forwarded on all the other output ports of the bridge. This action is referred to as **flooding** since it ensures that a copy of each frame transmitted is received on all segments in the total LAN. During the learning phase this procedure is repeated for each frame received by the bridge. In this way, all bridges in the LAN rapidly build up the contents of their forwarding databases.

This procedure works satisfactorily as long as stations are not allowed to migrate around the network (change their point of attachment) and the overall LAN topology is a simple tree structure (that is, there are no duplicate paths (routes) between any two segments). Such a tree structure is known as a **spanning tree**. Since in many networks, especially large networks, both these possibilities may occur, the basic learning operation is refined as follows.

The MAC address associated with a station is fixed at the time of its manufacture. If a user changes the point of attachment to the network of his or her PC/workstation, the contents of the forwarding database in each bridge must be periodically updated to reflect such changes. To accomplish this, an **inactivity timer** is associated with each entry in the database. Whenever [illegible] frame is received from a station within the predefined time interval, the timer expires and the entry is removed. Whenever a frame is received from a station for which the entry has been removed, the learning procedure is again followed to update the entry in each bridge with the (possibly new) port number. In this way the forwarding database in a bridge is continuously updated to reflect the current LAN topology and the addresses of the stations that are currently attached to the segments it interconnects. The inactivity timer also limits the size of the database since it contains only those stations that are currently active. This is important since the size of the database influences the speed of the forwarding operation.

The learning process works only if the total bridged LAN has a simple (spanning) tree topology. This means that there is only a single path between any two segments in the network. However, this condition may not always be met since additional bridges may be used to link two segments, for example, to improve reliability, or perhaps by mistake when a LAN is being updated.

Multiple paths between two segments cannot exist with the basic learning algorithm we have outlined since the flooding operation during the learning phase would cause entries in the forwarding database to be continuously overwritten. We can see this by considering the simple LAN topology shown in Figure 8.13. Clearly, if station 10 transmits a frame on segment 1 during the learning phase, then bridges B1 and B2 will both create an entry in their forwarding database – station 10/port 1 – and forward a copy of the frame onto segment 2. Each of these frames will in turn be received by the other bridge, and an entry will be made of station 10/port 2 and a copy of the frame output at port 1. In turn, each of these will be received by the other bridge, resulting in their corresponding entry for port 1 being updated. The frame will thus continuously circulate in a loop with the entries for each port being continuously updated.

Consequently, for topologies which offer multiple paths between stations, we need an additional algorithm to select just a single bridge for forwarding frames between any two segments. As we shall explain, this is done by setting

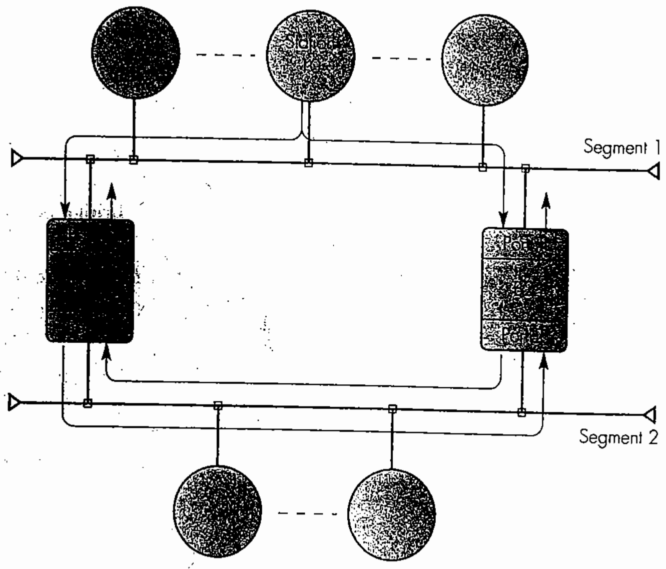


Figure 8.13 Effect of dual paths on learning algorithm.

selected bridge ports into the non-forwarding state and, in the topo
Figure 8.13, port 2 of bridge 2 would be selected. The resulting log
active topology behaves as a single spanning tree and the algorithm is
as the **spanning tree algorithm**. Note that although the algorithm selec
a single port/bridge for connecting two segments – making redunda
alternative bridges that may have been introduced to improve reliabili
example – the algorithm is run at regular intervals and will dynamically
a set of bridges from those currently operational.

Spanning tree algorithm

With the spanning tree algorithm, all the bridges regularly exchange sp
frames (messages), known as **bridge protocol data units (BPDUs).**
bridge has a priority value and a unique identifier. For the total bridged
a single bridge is dynamically chosen by the spanning tree algorithm
the **root bridge**. This is the bridge with the highest priority and the sma
identifier. It is determined/confirmed at regular intervals.

All the bridges in a LAN have a unique MAC group address which is
for sending all BPDUs between bridges. Also, the path cost associated
each bridge port is determined by the bit rate – known as the **designated**
– of the segment to which it is attached; the higher the bit rate, the sma
the designated cost. All bridges know the designated cost of the segmen
which they are attached.

When a bridge is first brought into service, it assumes it is the
bridge. A bridge that believes it is the root (all initially), initiates the trans
sion of a **configuration BPDU** on all of its ports (and hence the segm
connected to it) at regular time intervals known as the **hello time**.

Each configuration BPDU contains a number of fields including:

- the identifier of the bridge which the bridge transmitting the BPDU believes to be the root (itself initially);
- the path cost to the root from the bridge port on which the BPDU w received (zero initially);
- the identifier of the bridge transmitting the BPDU;
- the identifier of the bridge port from which the BPDU was transmit

On receipt of a configuration BPDU, each bridge connected to th
ment on which it was transmitted can determine, by comparing
identifier contained within it with its own identifier, firstly, whethe
bridge has a higher priority or, if the priorities are equal, whether its
identifier is less than the identifier from the received frame. If this is
case, the receiving bridge will carry on assuming it is the root and simply
card the received frame.

Alternatively, if the identifier from the received BPDU indicates that
not the root, the bridge proceeds by adding the path cost associated with

port on which the BPDU was received to that already within the frame. It then creates a new configuration BPDU containing this information, together with its own identifiers (bridge and port), and forwards a copy on all its other ports. This procedure is repeated by all bridges in the LAN. In this way the configuration BPDUs flood away from the root bridge to the extremities of the network and, at the same time, the path cost associated with each port of all the other bridges back to the root is computed. Thus, in addition to a single root bridge being established, all the other bridges will have determined the path cost associated with each of their ports. This is known as the **root path cost (RPC)** and the port that has the smallest RPC is then selected as the **root port** of the bridge. If two ports have the same RPC, the one with the lowest port number is chosen.

Once the root bridge and the root ports for the remaining bridges have been determined, the basis of the spanning tree has been established. The next step is to ensure that there are no loops/connections between the branches of the tree. This is done by selecting only a single bridge (port) to forward frames from each segment. This is known as the **designated bridge**. Its selection is based on the least path cost to the root bridge from the segment under consideration. If two bridge ports connected to a segment have the same path cost, the bridge with the smaller identifier is chosen. The bridge port connecting the segment to its designated bridge is known as the **designated port**. In the case of the root bridge, this is always the designated bridge for all the segments to which it is connected. Hence all its ports are designated ports.

When establishing the designated bridge port to be used with a segment, note that once a bridge port has been selected as a root port, it will not take part in the arbitration procedure to become a designated port. The choice of designated port is thus between the non-root ports connected to the segment under consideration.

The exchange of the configuration BPDUs between the two (or more) bridges involved will allow the two (or more) bridges to make a joint decision as to the port to be selected.

After the root bridge and the root and designated ports of all other bridges have been established, the state of the bridge ports can be set either to **forwarding** or to **blocking**. Initially, since all ports of the root bridge are designated ports, they are set to the forwarding state. For all the other bridges, only the root and designated ports are set to the forwarding state; the others are set to the blocking state. This establishes an active topology equivalent to a spanning tree. This procedure is then repeated at regular intervals (determined by the hello timer) to confirm the active topology or, in the event of a bridge failure, to reestablish a new topology.

xample 8.2

To illustrate how the various elements of the spanning tree algori
work, consider the bridged LAN shown in Figure 8.14(a). The uniq
identifier of each bridge is shown inside the box representing
bridge together with the port numbers in the inner boxes connecti
the bridge to each segment. Typically, the additional bridges on ea
segment are added to improve reliability in the event of a bridge f
ure. Also assume that the LAN is just being brought into service
bridges have equal priority and all segments have the same designat
cost (bit rate) associated with them. Determine the active (spannin
tree) topology.

Answer:

(i) First the exchange of configuration BPDUs will establish bridge
as the root bridge since this has the lowest identifier.

(ii) After the exchange of configuration BPDUs, the root path co
(RPC) of each port will have been computed. These are shown i
Figure 8.14(b).

(iii) The root port (RP) for each bridge is then chosen as the port wi
the lowest RPC. For example, in the case of bridge B3, port 1 ha
an RPC of 1 and port 2 an RPC of 2, so port 1 is chosen. In the ca
of B2, both ports have the same RPC and hence port 1 is chose
since this has the smaller identifier. The selected RPs are als
shown in the figure.

(iv) B1 is the root bridge so all its ports have a designated port co
(DPC) of 0. Hence they are the designated ports for segments S
S2 and S3.

(v) For S4 port 1 of B5 is an RP and hence is not involved in selecti
the designated port. The two other ports connected to S4 be
have a DPC of 1. Hence port 2 of B3 is selected as the designat
port because it has a lower identifier than B4.

(vi) For S5, the only port connected to it is port 2 of B5 and hence
is selected.

(vii) Finally, for S6 both ports have a DPC of 1, so port 2 of B
selected rather than port 2 of B6.

The DPCs are shown in Figure 8.14(c) and the resulting active topolo
is thus as shown in Figure 8.14(d). Note that the DPC of a port is alw
equal to the RPC of the root port of the bridge.

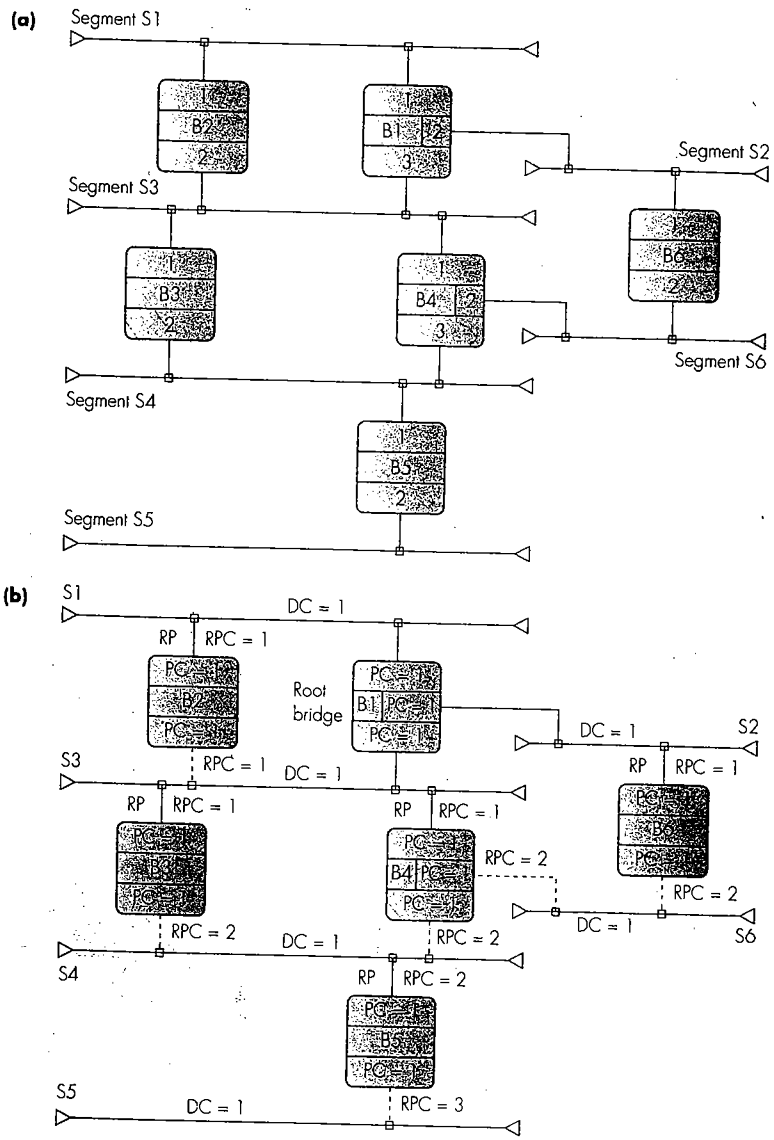


Figure 8.14 Active topology derivation example: (a) LAN topology; (b) root port selection; (c) designated port selection; (d) active topology.

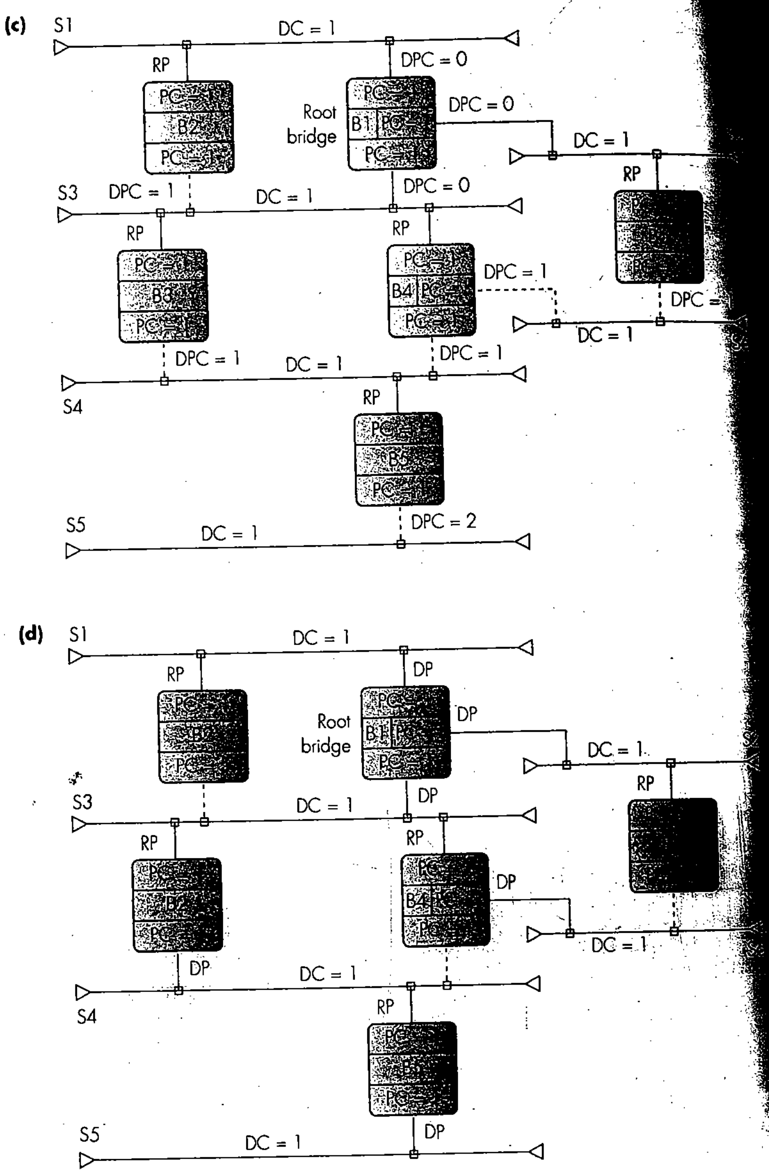

(c)
S1
DC = 1
RP
DPC = 0
Root bridge
B1
B2
DPC = 0
DC = 1
RP
S3
DPC = 1
DC = 1
DPC = 0
RP
RP
B3
B4
DPC = 1
DPC =
DC = 1
DPC = 1
DC = 1
DPC = 1
S4
RP
B5
S5
DC = 1
DPC = 2
(d)
S1
DC = 1
RP
DP
Root bridge
B1
DP
DC = 1
RP
S3
DC = 1
DP
RP
RP
B4
DP
DC = 1
DP
DC = 1
S4
RP
S5
DC = 1
DP

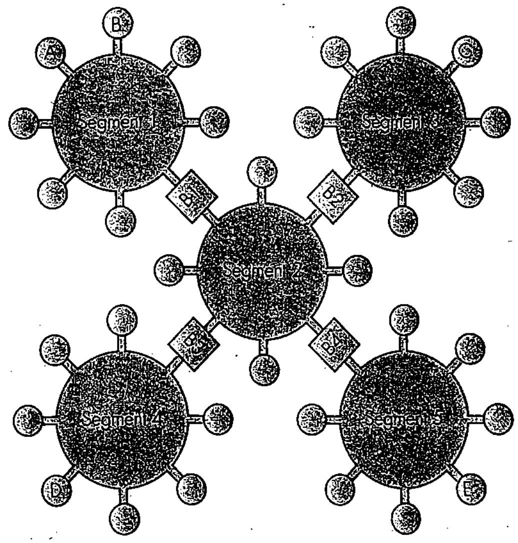
Figure 8.14 Continued.

8.5.2 Source routing bridges

Although we can use **source routing bridges** with any type of LAN segment, we use them primarily for the interconnection of token ring LAN segments. A typical network based on source routing bridges is shown in Figure 8.15(a).

The major difference between a LAN based on source routing bridges and one based on spanning tree bridges is that with the latter the bridges collectively perform the routing operation in a way that is transparent to the end stations. Conversely, with source routing, the end stations perform the routing function.

(a)


A
B
C
D
E
Segment 1
Segment 3
Segment 2
Segment 4
Segment 5
B1
B2
B3
B4

(b) Routing table held by A: Destination B = Segment 1 (same segment)
C = Segment 1, B1, Segment 2, B2, Segment 3
⋮
E = Segment 1, B1, Segment 2, B4, Segment 5

Routing table held by B: Destination A = Segment 1 (same segment)
⋮
D = Segment 1, B1, Segment 2, B3, Segment 4

Routing table held by E: Destination A = Segment 5, B4, Segment 2, B1, Segment 1
⋮
C = Segment 5, B4, Segment 2, B2, Segment 3

Figure 8.15 An example source routing bridged LAN: (a) topology; (b) routing table entries.

With source routing, a station ascertains the route to be followed by a frame to each destination before any frames are transmitted. This information is inserted at the head of the frame and is used by each bridge to determine whether a received frame is to be forwarded on another segment or not. The routing information comprises a sequence of segment-bridge, segment-bridge identifiers. Routing tables for selected stations in the example network are shown in Figure 8.15(b).

On receipt of each frame, a bridge needs only to search the routing field at the head of the frame for its own identifier. Only if it is present and followed by the identifier of a segment connected to one of its output ports does it forward the frame on the specified LAN segment. Otherwise it is not forwarded. In either event, the frame is repeated at the ring interface by the bridge and, if forwarded, the address-recognized (A) and frame-copied (C) bits in the frame status (FS) field at the tail of the frame are set to indicate to the source station (bridge) that is has been received (forwarded) by the destination station (bridge).

Routing algorithm

The routing information field contained within each frame immediately follows the source address field at the head of the normal information frame. The modified frame format is thus as shown in Figure 8.16(a).

Since a routing information field is not always required – for example, if the source and destination stations are on the same segment – the first bit of the *source address* – the individual/group (I/G) address bit – is used to indicate whether routing information is present in the frame (1) or not (0). This can be done since the source address in a frame must always be an individual address, so the I/G bit is not needed for this purpose.

If routing information is present, its format is as shown in Figure 8.16(b). The *routing information* field consists of a *routing control* field and one or more *route designator* fields. The routing control field itself comprises three subfields: *frame type, maximum frame size,* and *routing field length.* In addition to normal information frames, two other frame types are associated with the routing algorithm. The frame type indicates the type of the frame.

Source routing bridges can be used for the interconnection of different types of LAN segments in addition to token rings. Since there is a different maximum frame size associated with each segment type, the *maximum frame size* field determines the largest frame size that can be used when transmitting a frame between any two stations connected to the LAN.

Prior to transmitting a route finding frame, a station sets the maximum frame size field to the (known) largest frame size that can be used in the total LAN. Before a bridge forwards the frame on a segment, the bridge checks this field with the (known) maximum frame size of the new segment. If the latter is smaller, the bridge reduces the frame size field to the lower value. In this way, the source station, on receipt of the corresponding route reply frame, can use this information when preparing frames for transmission to that destination.

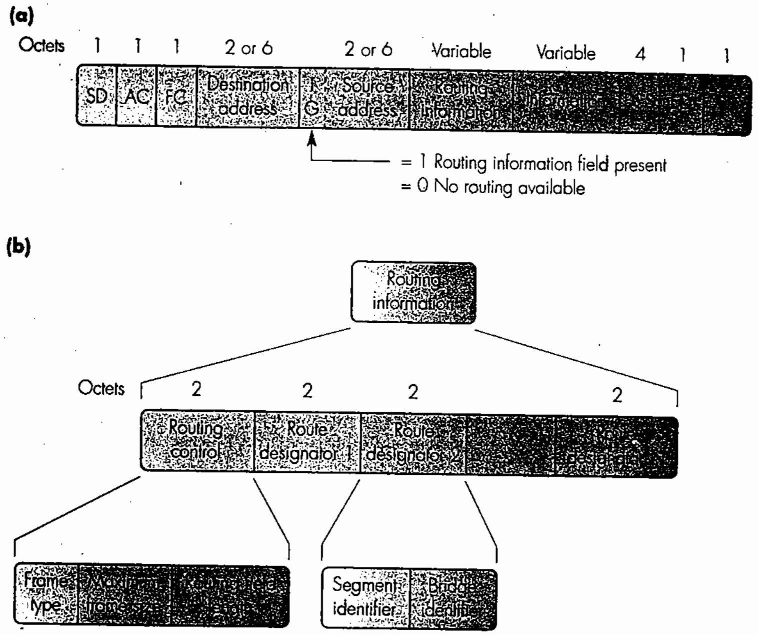


Figure 8.16 Token ring frame format: (a) position of routing information field; (b) structure of routing information field.

Since the number of segments (and bridges) traversed by a frame when going from a source to a destination may vary, the routing field length indicates the number of route designators present in the rest of the routing information field. Each route designator comprises a pair of segment and bridge identifiers.

The two additional frame types associated with the route finding algorithm are the **single-route broadcast frame** and the **all-routes broadcast frame**. To find a route, a station first creates and transmits a single-route broadcast frame with a zero routing field length and the maximum frame size set to the known largest value for the total LAN. As with spanning tree bridges, source routing bridges operate in the promiscuous mode and hence receive and buffer all frames at each of their ports. On receipt of a single-route broadcast frame, a bridge simply broadcasts a copy of the frame on each of the segments connected to its other ports. Since this procedure is repeated by each bridge in the LAN, a copy of the frame propagates throughout the LAN and is thus received by the intended destination station irrespective of the segment on which it is attached.

As we indicated earlier in Figure 8.13, if there are redundant (and hence loops) in the LAN topology, multiple copies of the fra propagate around the LAN. To prevent this, before any route finding are sent, the bridge ports are configured to give a spanning tree activ ogy. On the surface, this may appear to be the same procedure us transparent bridges. With source routing bridges, however, the re spanning tree active topology is used only for routing the initial singl broadcast frames. This ensures that only a single copy of the frame pro gates through the network. The spanning tree active topology is not used routing either normal information frames or the all-routes broadcast fram

On receipt of a single-route broadcast frame, the required destina station returns an all-routes broadcast frame to the originating sta Unlike the single-route broadcast, this frame is not constrained to follo spanning tree active topology at each intermediate bridge. Instead, receipt of such frames, the bridge simply adds a new route designator (comprising the segment identifier on which the frame was received and own bridge identifier), increments the routing field length, and then broa casts a copy of the frame on each of its other port segments.

In this way, one or more copies of the frame will be received by the o nating source station via all the possible routes between the two stations. examining the route designators in their routing control fields, the sou station can select the best route for transmitting a frame to that destinatio This route is then entered into its routing table and is subsequently u when transmitting any frames to that station.

Since the all-routes broadcast frame is not constrained to follow the sp ning tree active topology, on receipt of such frames additional steps must taken by each bridge to ensure that no frames are simply circulating in loop Before transmitting a copy of the all-routes broadcast frame on an out segment, each bridge first searches the existing routing information in frame to determine if the segment identifiers associated with the incom and outgoing ports are already present together with its own bridge ide fier. If they are, a copy of the frame has already been along the route, so copy of the frame is not transmitted on the segment.

Note that it is not necessary to perform the route finding operation each frame transmitted. Once a route to an intended destination has be determined and entered (cached) into the routing table of a station, this be used for the transmission of all subsequent frames to that destinatio Moreover, since most stations transmit the majority of their frames to a li ited number of destinations, the number of route finding frames is relative small compared with normal information frames for modest sized LANs.

Example 8.3

Assume the bridged LAN shown in Figure 8.17(a) is to operate using source routing. Also assume that all bridges have equal priority and all rings have the same designated cost (bit rate). Derive the following when station *A* wishes to send a frame to station *B*:

(i) the active spanning tree for the LAN,

(ii) all the paths followed by the single-root broadcast frame(s),

(iii) all the paths followed by the all-routes broadcast frame(s),

(iv) the route (path) selected by *A*.

Answer:

(i) (a) Bridge B1 has the lowest identifier and is selected as the route.

(b) The root ports for each bridge are then derived as shown.

(c) The designated ports for each segment can now be derived and these are as shown.

(d) The active topology is as shown in Figure 8.17(b).

(ii) Paths of single-route broadcast frames:

R1→B1→R2→B2→R3
R2→B3→R5→B6→R6
R1→B4→R4→B5

(iii) Paths of all-routes broadcast frames:

R6→B6→R5→B3→R2→B2→R3
B2→B1→R1
B1→B4→R4→B5→R5→B3

(iv) Since each ring has the same bit rate, the route (path) selected is either:

R1→B1→R2→B3→R5→B6→R6
or
R1→B4→R4→B5→R5→B6→R6

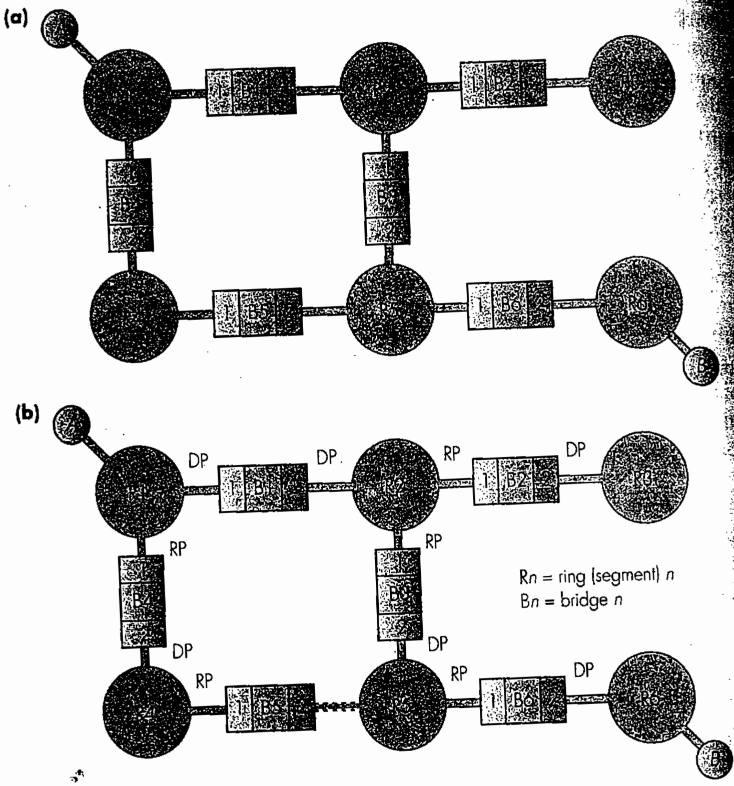
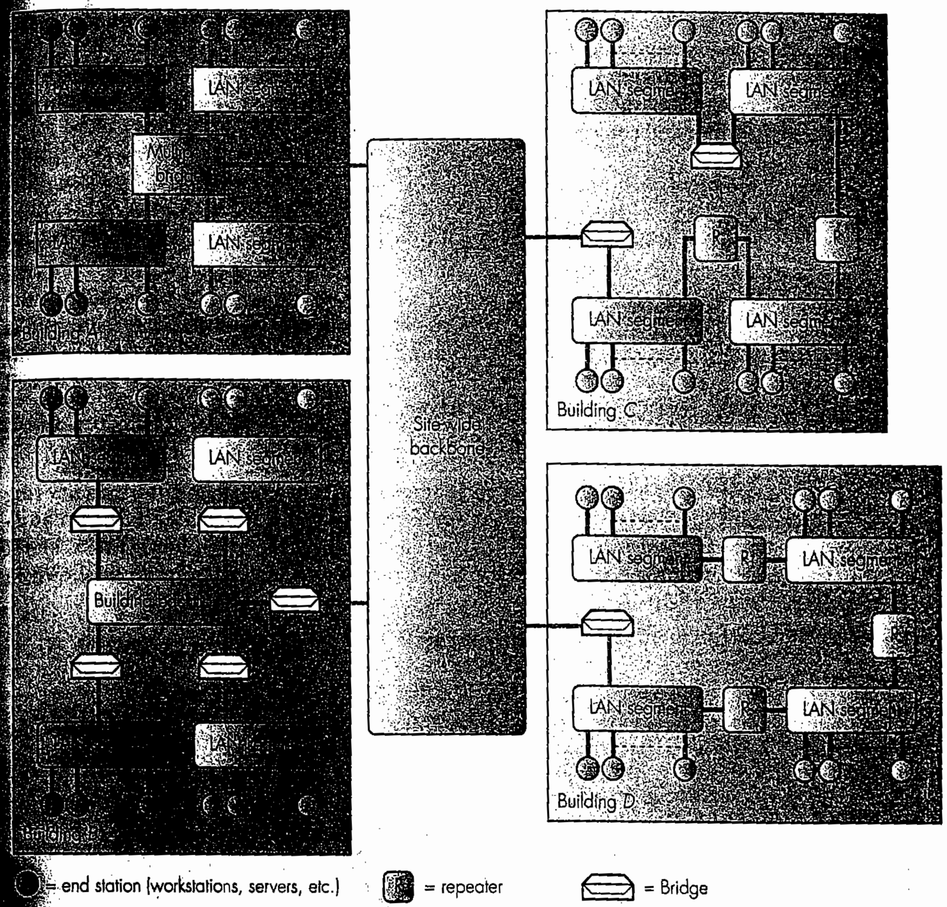

(a)
1 B1
1 B2
B3
B4
1 B5
1 B6
R6
(b)
DP
RP
R1
R2
R3
R4
R5
R6
1 B1
1 B2
B3
B4
1 B5
1 B6
Rn = ring (segment) n
Bn = bridge n

Figure 8.17 Source routing example: (a) topology; (b) spanning tree.

8.6 FDDI

As we explained in Section 8.2, in addition to bridges being introduced as means of avoiding the traffic that is local to a segment from unnecessaril loading the entire LAN, backbone subnetworks were introduced to ensur the traffic that was forwarded between segments incurred only minimal delays. As an example, a small (but typical) establishmentwide LAN tha includes bridges and backbone subnetworks is given in Figure 8.18.

Normally no end systems (workstations, servers, and so on) are connected to a backbone and they are used solely for intersegment traffic. For


LAN segment
Multi bridge
Building A
Building B
Building C
Building D
Site-wide backbone
= end station (workstations, servers, etc.)
= repeater
= Bridge

ure 8.18 Typical establishmentwide LAN.

interconnecting only a small number of segments in a single building, backbones of the same type as the interconnected segments (CSMA/CD, or token ring) are used. They are known as **building backbones**.

As the number of interconnected segments increases, there comes a point at which the transmission bandwidth required by the backbone to meet the intersegment traffic starts to exceed that available with the basic LAN types. To overcome this problem, backbones based on newer high-speed LAN types are used. An example is the **fiber distributed data interface (FDDI) LAN**. This is an optical fiber-based ring network that supports a bit rate of

100 Mbps. It can be used for the interconnection of segments spread over a wider geographical area than a single building, such as a university campus or manufacturing plant. The resulting network is then known as an **establishment or site backbone**. We explain the principle of operation of FDDI in this section and some other high-speed LANs in Sections 8.7 and 8.8.

The FDDI LAN standard was developed by the American National Standards Institute (ANSI). It is now an international standard and is defined in ISO 9314. It is based on a ring topology and operates at a bit rate of 100 Mbps. Because of its role, it uses dual counter-rotating rings to enhance reliability. Multimode fiber connects each station together and the total ring can be up to 100 km in length. Up to 500 stations can be connected in the ring and hence it forms an ideal backbone network. The MAC method is based on a control token and, in addition to normal data traffic, optionally the ring can also support delay-sensitive traffic that requires a guaranteed maximum access delay; for example, digitized speech and video.

Network configuration

FDDI uses two counter-rotating rings to enhance reliability: the **primary ring** and the **secondary ring**. The secondary ring can be used either as an additional transmission path or purely as a back-up in the event of a break occurring in the primary ring. A typical network configuration is shown in Figure 8.19.

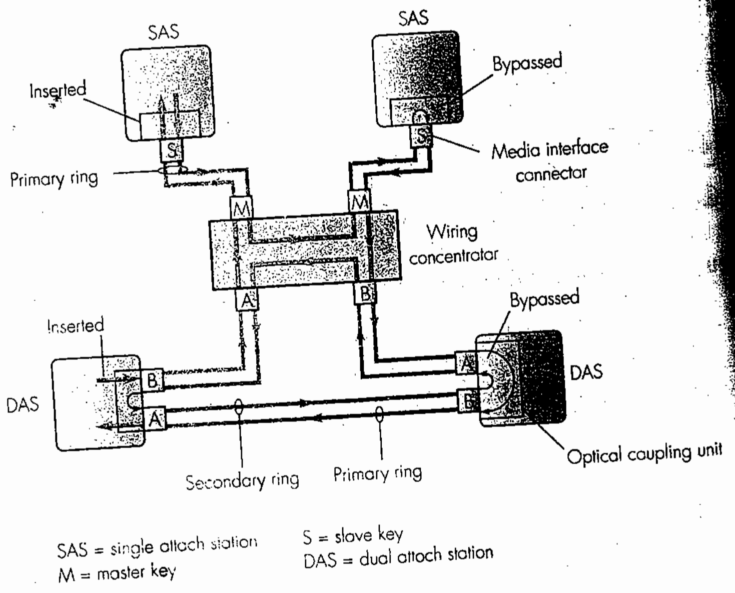


Figure 8.19 FDDI networking components.

As we can see, there are two types of station: a **dual attach station (DAS)** which is connected to both rings and a **single attach station (SAS)** which is attached only to the primary ring. In practice, most user stations are attached to the ring via **wiring concentrators** since then only a single pair of fibers is needed and the connection cost is lower.

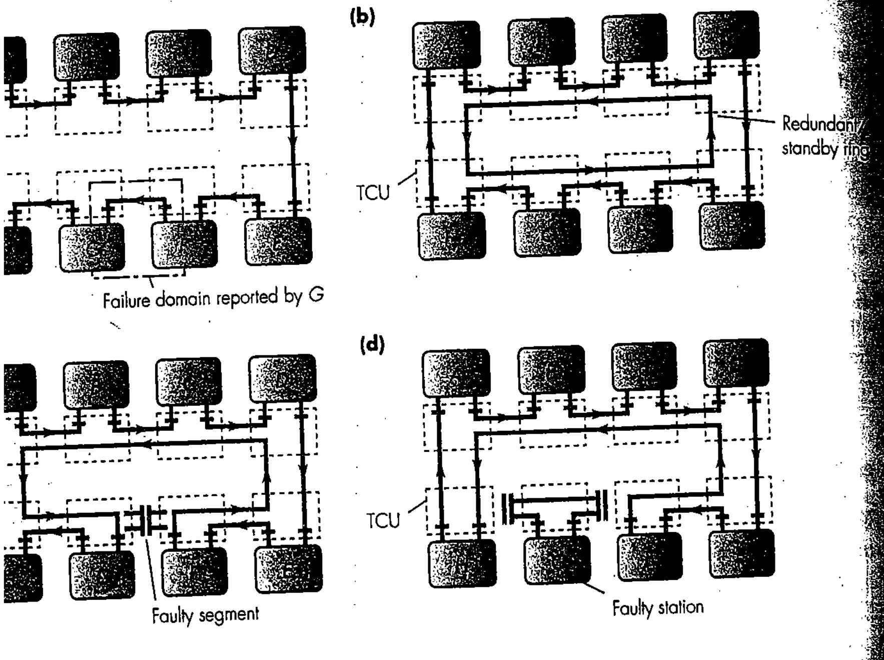
If the LAN is being used as a backbone, then most attached stations are bridges. The protocol used to reconfigure the LAN into a single ring in the event of a ring failure is the same as the beaconing procedure associated with a token ring LAN. As we explained in Section 8.4, if a serious failure such as a broken cable arises in the ring, beacon MAC frames are issued by the station that detects the failure. Essentially, these are used to inform all other stations in the ring that the token-passing protocol has been suspended until the affected failure domain has been located and repaired. The failure domain consists of the station that detects the failure, its immediate upstream neighbor, and the ring segment in between them. The failure has deemed to be repaired when the station that issues the beacon frames starts to receive them after rotating around the ring; that is, on receipt of a beacon frame with its own MAC address at the head of the frame.

An example of a failure domain is shown in Figure 8.20(a) and a redundant ring configuration in Figure 8.20(b). In this example we assume a break has occurred in the ring segment between stations *F* and *G*. Hence *G* is the beaconing station and *F* its upstream neighbor. When a redundant ring is being used, the TCU not only supports the functions we explained in Section 8.4, but also the means to bypass a faulty ring segment or station. As an example, we show in Figure 8.20(c) how the faulty ring segment (failure domain) we identified in Figure 8.20(a) is bypassed.

Essentially, once the failure domain has been located and reported, the relays in the TCU of *F* and *G* are activated to (hopefully) re-establish a continuous ring. If isolating the suspected faulty segment does not remove the fault, the next step is to initiate the isolation of station *G* completely, as shown in Figure 8.20(d). Note from these illustrations that the redundant ring does not have a direct path to the MAC unit and simply provides a means of bypassing a section of the ring. The order of the stations in a re-established ring is the same as that in the original ring.

The basic fiber cable is **dual core** with **polarized duplex** (two-position) connectors at each end. This means that each end of the cable has a different physical key so that it can be connected into a matching socket only. This prevents the transmit and receive fibers from becoming inadvertently interchanged and bringing down the total network. As a further precaution, we use different connectors to connect each station type – SAS and DAS. In common with the basic token ring, we use special coupling units to isolate (bypass) a station when its power is lost. With FDDI, these are either active or passive fiber devices.

Although the topology is logically a ring, physically it is normally implemented in the form of a hub/tree structure. An example is shown in Figure 8.21(a). To ensure that changes to the wiring are carried out in a


(b)
Redundant/
standby ring
TCU
Failure domain reported by G
(d)
TCU
Faulty segment
Faulty station

nk coupling units

.20 Ring fault detection and isolation: (a) failure detection; (b) redundant ring
ration; (c) segment isolation; (d) station isolation.

controlled way, a combination of **patch panels** and **wiring concentrators** ar
used. Typically, these are located in the wiring closet associated with either a
floor (if a local FDDI ring is being used) or a building (if the ring is a bac
bone). In the latter case, the patch panels are interconnected as shown i
Figure 8.21(b) to create a tree structure to connect each station – high-spee
servers or bridges – to the ring.

Each patch panel has a number of possible attachment points associate
with it. In the absence of a connection at a particular point, the ring is mai
tained using short **patch cables**, each with the same type of connecto
Adding a new station or concentrator simply involves removing a patch cab
and replacing it with a corresponding drop cable. This approach is known a
structured wiring.

Physical interface

The physical interface to the fiber cable is shown in Figure 8.22. In a basi
token ring network, at any instant, there is a single active ring monitor which

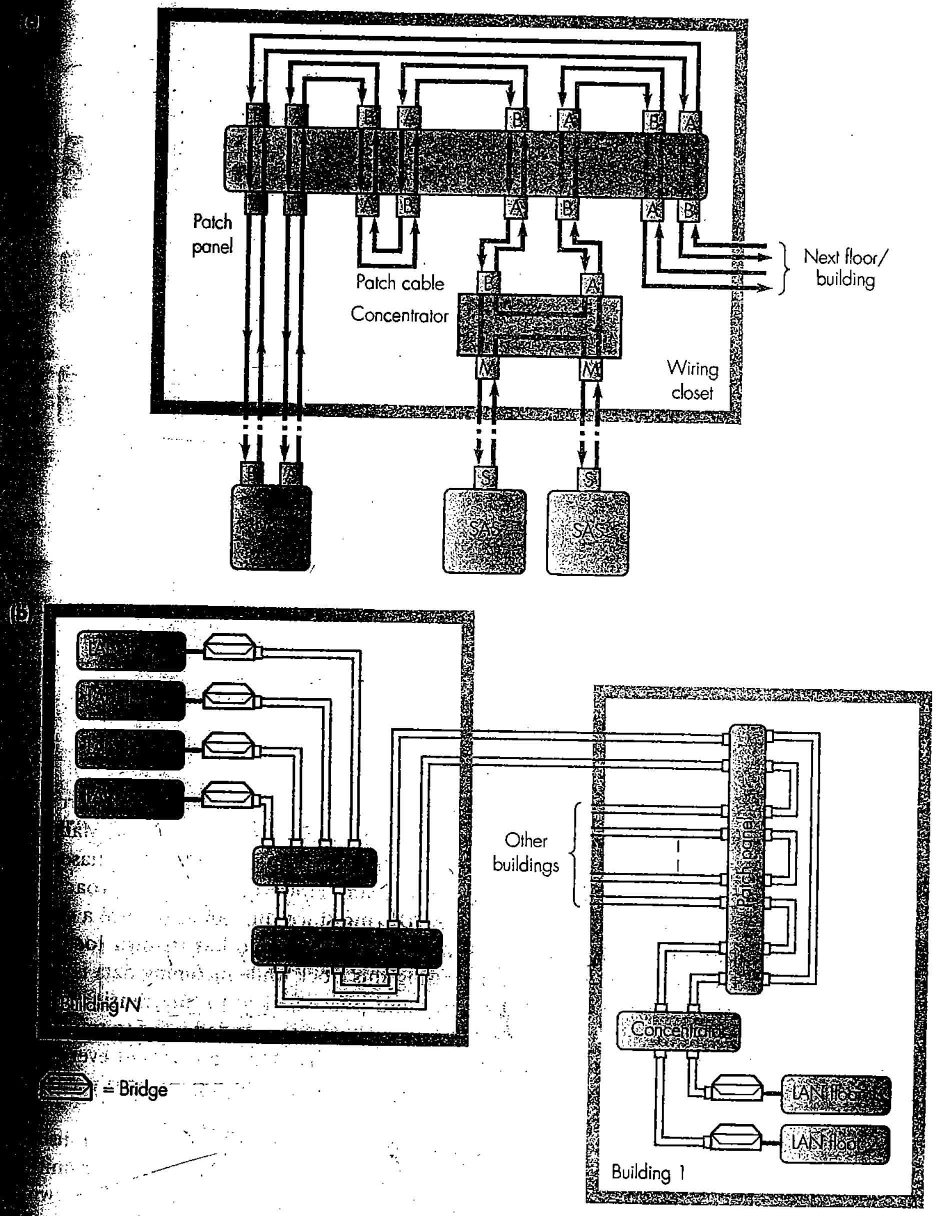

Patch panel
B
A
Patch cable
Concentrator
M
Wiring closet
Next floor/ building
S
SAS
SAS
(b)
Other buildings
Building N
Patch panel
Concentrator
LAN floor
LAN floor
Building 1
= Bridge

Figure 8.21 FDDI wiring schematic: (a) building; (b) establishment.

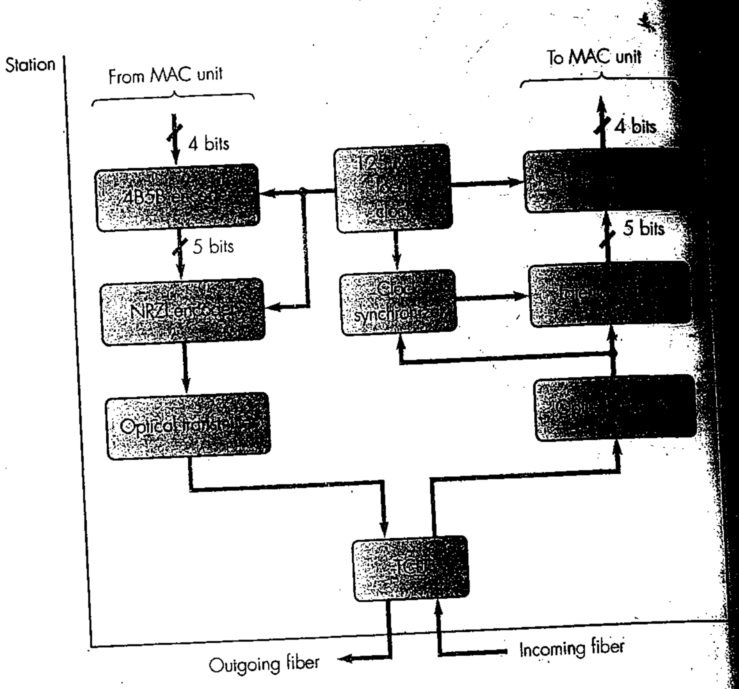


Figure 8.22 FDDI physical interface schematic.

among other things, supplies the master clock for the ring. Each circula
bitstream is encoded by the active ring monitor using differential Manche
encoding. All the other stations in the ring then frequency and phase lo
the clock extracted from this bitstream. However, such an approach
suitable at the bit rates of an FDDI ring since this would require a sig
rate of 200 Mbaud. Instead, each ring interface has its own local
Outgoing data is transmitted using this clock while incoming data is re
using a clock that is frequency and phase locked to the transitions
incoming bitstream. As we shall see, all data is encoded prior to transmis
so that there is a guaranteed transition in the bitstream at least every two
cell periods, ensuring that each received bit is sampled (clocked) very nea
the nominal bit cell center.

All data to be transmitted is first encoded, prior to transmission, using
of 5 group code. This means that for each 4 bits of data to be transmitted
corresponding 5-bit codeword or symbol is generated by what is known a
4B5B encoder. The 5-bit symbols corresponding to each of the sixteen possib
4-bit data groups are shown in Figure 8.23(a). As we can see, there is a ma
mum of two consecutive zero bits in each symbol. The symbols are then shifte

(a)

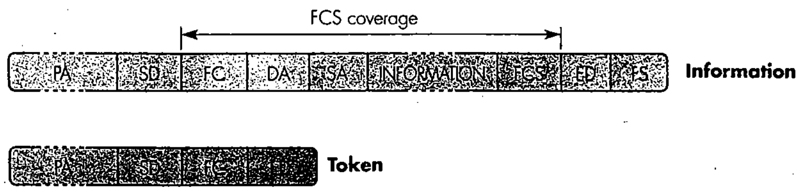
Data symbols

4-bit data group	5-bit symbol
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Control symbols

IDLE ------ 11111
J ------ 11000
K ------ 10001
T ------ 01101
R ------ 00111
S ------ 11001
QUIET ------ 00000
HALT ------ 00100

(b)



PA = preamble (16 or more symbols)
SD = start delimiter (2 symbols)
FC = frame control (2 symbols)
DA = destination address (4 or 12 symbols)
SA = source address (4 or 12 symbols)
FCS = frame check sequence (8 symbols)
ED = end delimiter (1 or 2 symbols)
FS = frame status (3 symbols)

igure 8.23 FDDI line coding and framing detail: (a) 4B5B codes; (b) frame formats.

out through a further NRZI encoder which produces a signal transition whenever a 1 bit is being transmitted and no transition when a 0 bit is transmitted. In this way, there is a guaranteed signal transition at least every two bits.

The use of 5 bits to represent each of the sixteen 4-bit data groups means that there are a further sixteen unused combinations of the 5 bits. Some of these combinations (symbols) are used for other (link) control functions, such as indicating the start and end of each transmitted frame or token. A list of the link control symbols is shown in Figure 8.23(a) and part (b) shows the format used for frames and tokens. In general, the meaning and use of each field is the same as with the basic token ring but, because symbols are used rather than bits, there are some differences in the structure of each field.

The *preamble* (*PA*) field consists of 16 or more IDLE symbols which, since they each consist of five 1 bits, cause the line signal to change at the maximum frequency. The line signal transitions are used for establishing (and maintaining) clock synchronization at the receiver. The *start delimiter* (*SD*) field consists of two control symbols (J and K) which enable the receiver to interpret the following frame contents on the correct symbol boundaries. The *FC*, *DA*, and *SA* fields have the same meaning as before, but the (decoded) information field in the data frames can be up to 4500 octets with FDDI. The *end delimiter* (*ED*) field contains one or two control symbols (*T*). Finally, the frame status (*FS*) field, although it has a similar function to the FS field in the basic ring, consists of three symbols that are combinations of the two control symbols R and S.

The local clock used in the physical interface is 125 MHz which, because of 4B5B encoding, yields a data rate of 100 Mbps and a signaling rate of 125 Mbaud. Since all transmissions are encoded into 5-bit symbols, each 5-bit symbol must first be buffered at the receiver before it can be decoded. However, the use of two symbols (J and K) for the SD field to establish correct symbol boundaries means that a 10-bit buffer is used at the receiver. This is known as the **latency** (or elastic) **buffer** since it introduces 10 bits of delay – latency – into the ring. At 125 Mbaud, this is equivalent to a delay of 0.08 µs but, allowing for additional gate and register transfer delays, this is normally rounded up to 1 µs per ring interface.

mple 8.4

Assuming a signal propagation delay in the fiber of 5 µs per 1 km, derive the latency of the following FDDI ring configurations in both time and bits assuming a usable bit rate of 100 Mbps:

(i) 2 km ring with 20 stations,

(ii) 20 km ring with 200 stations,

(iii) 100 km ring with 500 stations.

Answer:

Ring latency, T_l = Signal propagation delay T_p + N × station latency T_s where N is the number of stations.

(i) $T_l = 2 \times 5 + 20 \times 1$

$= 30\,\mu s$ or 3000 bits

(ii) $T_l = 20 \times 5 + 200 \times 1$

$= 300\,\mu s$ or 30000 bits

(iii) $T_l = 100 \times 5 + 500 \times 1$

$= 1000\,\mu s$ or 100000 bits

Note that the above values assume that the primary ring only is in use. If a fault occurred, the three signal propagation delay values would each be doubled. Also, for dual attach stations, the station latency would be doubled.

Frame transmission and reception

The short latency time (and hence number of bits in circulation) of the basic token ring means that a station, after initiating the transmission of an information frame, can wait until the FS field at the tail of the frame has been received before transmitting a new token without any noticeable loss in ring utilization. However, from Example 8.4 we can deduce that with an FDDI ring the loss in ring utilization will be significant if this mode of operation is adopted. With an FDDI ring, therefore, the early token release method is utilized; that is, a new token is transmitted immediately after the station has transmitted the FS symbol at the tail of a frame. The station then follows the token with IDLE symbols until it receives the SD symbols indicating the start of a new frame or token. The basic scheme is shown in Figure 8.24.

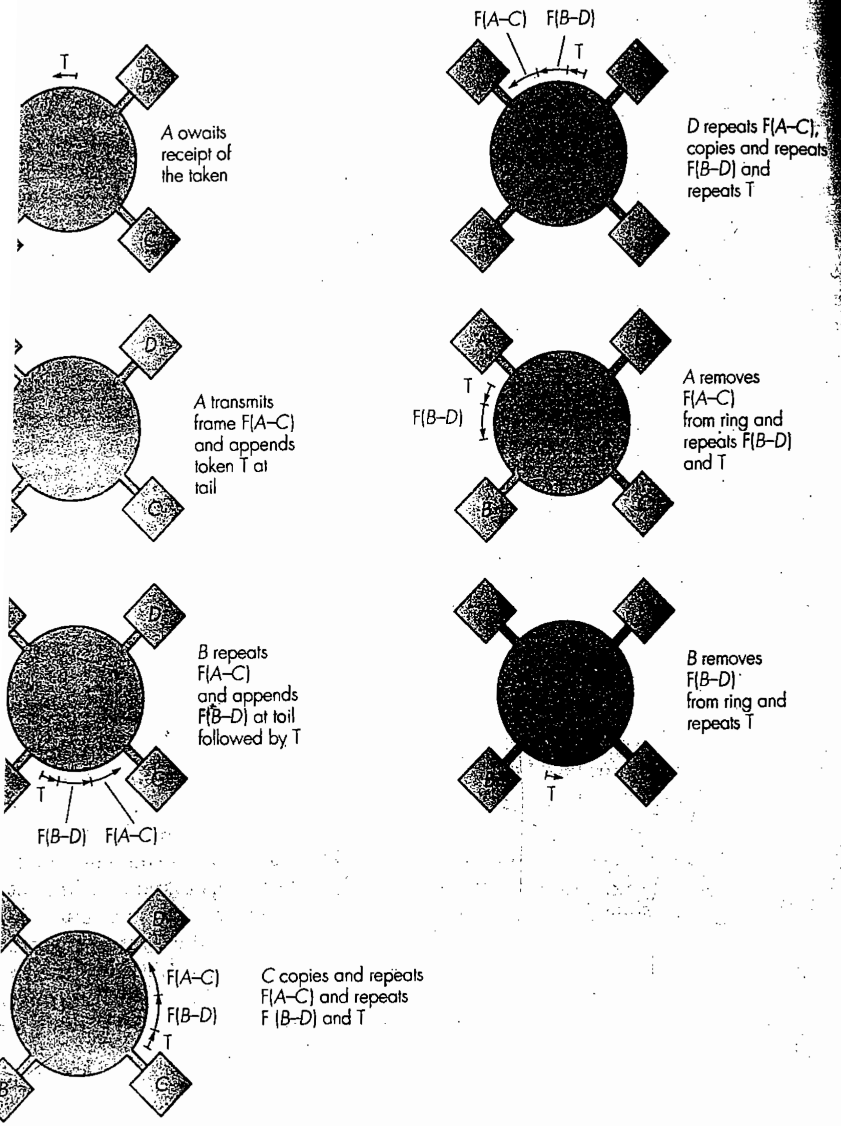
As we can see, as with the basic token ring, the source station removes a frame after it has circulated the ring. However, because of the long latency of an FDDI ring, more than one frame may be circulating around the ring at one time. Although not shown in Figure 8.24, the ring interface must repeat the SD, FC, and DA fields (symbols) of any received frames before it can determine if its own address is in the SA field. This can result in one or more frame fragments – comprising SD, FC, and DA fields – circulating around the ring. This means that a station, on receipt of the token, starts to transmit a waiting frame and concurrently receives and discards any frame fragments that may be circulating around the ring.

Timed token rotation protocol

Unlike the transmission control method used with the basic token ring – which is based on the use of the priority and reservation bits in the access control field of token and information frames – an FDDI ring uses a scheme that is based on a timed token rotation protocol that is controlled by a preset parameter known as the **target token rotation time (TTRT)**.

For each rotation of the token, each station computes the time that has expired since it last received the token. This is known as the **token rotation time (TRT)** and includes the time taken by a station to transmit any waiting frames plus the time taken by all other stations in the ring to transmit any waiting frames for this rotation of the token. Clearly, if the ring is lightly loaded then the TRT is short but, as the loading of the ring increases, so the TRT measured by each station increases also. The TRT is thus a measure of the total ring loading.

The timed token rotation protocol ensures access to the ring is shared fairly between all stations by allowing a station to send any waiting frames only if its measured TRT is less than the preset TTRT for the ring. Hence, when a station has frames waiting to send, on receipt of the token it computes the difference between the TTRT and its current TRT. This is known as the **token hold time** (THT) since it determines for how long the station may transmit on A has a frame waiting to send to station C and station B has a frame waiting to send to station D


T
D
C
A awaits receipt of the token
A transmits frame F(A–C) and appends token T at tail
B repeats F(A–C) and appends F(B–D) at tail followed by T
T
F(B–D)
F(A–C)
C copies and repeats F(A–C) and repeats F (B–D) and T
F(A–C)
F(B–D)
T
F(A–C)
F(B–D)
T
D repeats F(A–C), copies and repeats F(B–D) and repeats T
T
F(B–D)
A removes F(A–C) from ring and repeats F(B–D) and T
B removes F(B–D) from ring and repeats T
T

24 **FDDI transmission example.**

waiting frames before releasing the token. If the THT is positive, then the station can transmit for up to this interval. If it is negative, then the station must forego transmitting any waiting frames for this rotation of the token. A positive THT is thus known as an **early token** while a negative THT is known as a **late token**.

Example 8.5

The timed token rotation protocol is to be used to control access to a four-station FDDI ring network. All frames to be transmitted are of the same length and the TTRT to be used is equivalent to the transmission of four frames plus any ring latency. After an idle period when no frames are ready to send, all four stations receive a block of frames to send. Assuming the time for the token to rotate around the ring in the idle state is equal to the time to transmit the token, T_t, plus the ring latency, T_l, derive and show in table form the number of frames each station can transmit for the next four rotations of the token.

Answer:

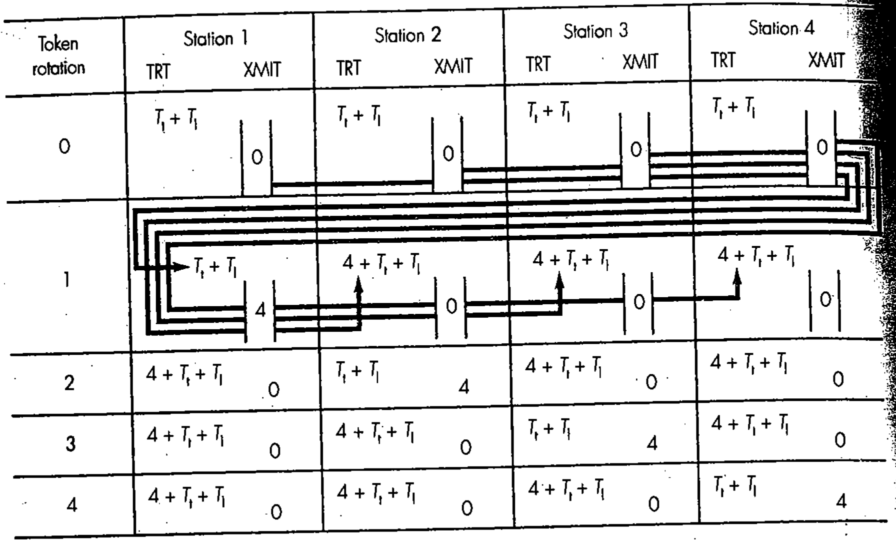
The number of frames that each station can transmit on the first four rotations of the token are shown in diagrammatic form in Figure 8.25. They are derived as follows:

After the idle period, the TRT of all stations will [illegible] since no frames have been transmitted. Once frames became [illegible] receipt of the token station 1 computes a THT of [illegible] (XMITs) four waiting frames before passing [illegible] since station 1 has sent four frames, none of the other stations [illegible] send any frames for this rotation of the token. This is [illegible] is now greater than 4 in each case and hence [illegible] puted THT is negative.

On the second rotation of the token, the TRT of station 1 has [illegible] ment [illegible] 4 plus the ring latency [illegible] waiting frames. This means that the TRT of station 2 is less than the TTRT and the computed THT is again 4 [illegible] frames. Again this will block stations 3 and station 4 [illegible] frames on this rotation of the token.

On the third rotation of the token, the THT of stations 1 and 2 [illegible] plus the ring latency and hence neither can send frames. The TRT of station 3, however, is this time less than the TTRT and the computed THT is 4 hence it can transmit four waiting frames. Again station four is blocked.

Finally, on the fourth rotation of the token, stations 1, 2, and 3 are still all blocked and station 4 can transmit four frames. This simple example shows that the available transmission capacity is shared in an equitable way between all four stations.



Token rotation	Station 1 TRT	Station 1 XMIT	Station 2 TRT	Station 2 XMIT	Station 3 TRT	Station 3 XMIT	Station 4 TRT	Station 4 XMIT
0	$T_t + T_l$	0	$T_t + T_l$	0	$T_t + T_l$	0	$T_t + T_l$	0
1	$T_t + T_l$	4	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0
2	$4 + T_t + T_l$	0	$T_t + T_l$	4	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0
3	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0	$T_t + T_l$	4	$4 + T_t + T_l$	0
4	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0	$4 + T_t + T_l$	0	$T_t + T_l$	4

TRT = token rotation time
XMIT = number of frames transmitted on this rotation of the token
TTRT = target token rotation time

T_t = time to transmit the token
T_l = ring latency
TTRT = $4 + T_t + T_l$

igure 8.25 FDDI timed token rotation protocol example.

Performance

There are two important performance measures associated with shar medium networks: the **maximum obtainable throughput** and the **maximu access delay**. Both are strongly influenced by the MAC algorithm use share the available transmission capacity between the attached stations. As have just seen, in the case of an FDDI ring this is based on the use of a co token and the timed token rotation protocol. The important parameter as ciated with the protocol is the TTRT and, since this must be preset to same value in all stations, it is important to quantify its effect on both obtainable throughput and the access delay.

Although the nominal throughput of an FDDI ring is 100 Mbps, becau of the ring latency and access control mechanism, the maximum obtainab throughput is less than this. Maximum obtained throughput (and acce delay) implies that there are always frames waiting to be sent on receipt of th token at each ring interface. Under such conditions, we can compute th obtainable maximum throughput by considering the operating scenario i Example 8.5. In this, on receipt of the token, the first station transmits a se of frames up to the point when the TTRT expires before passing on th

token. All the other stations in the ring are blocked from transmitting any frames during this rotation of the token. It is not until the token is passed to the next station in the ring on the following rotation that the second transmission burst of frames occurs.

As we can deduce from this, the transmission time lost during successive rotations of the token is made up of two parts: the time lost while the set of frames transmitted by the first station rotate around the ring – the ring latency – and the time taken for the token to be passed to the next station on the ring. We can express the maximum utilization of the nominal ring capacity, $U_{\max}$, as follows:

$$U_{\max} = \frac{\text{TTRT} - T_l}{(\text{TTRT} - T_l) + T_l + T_t + (T_l / n)} = \frac{n\,(\text{TTRT} - T_l)}{n\,(\text{TTRT} - T_l) + (n+1)\ T_l + T_t}$$

where TTRT is the target token rotation time, T_l is the ring latency, T_t is the time to transmit the token, and n is the number of stations in the ring.

In practice, the TTRT is very much greater than T_t. Hence the maximum utilization can be approximated to:

$$U_{\max} = \frac{n\,(\text{TTRT} - T_l)}{n\text{TTRT} - T_l} \quad \text{or approximately,} \quad \frac{\text{TTRT} - T_l}{\text{TTRT}}$$

We can conclude that, to achieve a high level of ring utilization, we must select a TTRT which is significantly greater than the total ring latency. In addition, the selected TTRT must allow at least one frame of the maximum size to be present on the ring. The maximum size frame is 4500 bytes which requires 0.36 ms to transmit at 100 Mbps. The maximum ring latency was derived in Example 8.4 and, allowing for the secondary ring to be in use and the 500 stations to be all dual-attach stations, this is approximately 2 ms. Therefore the minimum TTRT allowable is 2.36 ms which, allowing for a safety margin, is set to 4 ms in the standard.

The access delay is defined as the time delay between the arrival of a frame at the ring interface of the source station and its delivery by the ring interface at the destination station. Thus access delay includes any time spent waiting in the ring interface queue at the source until a usable token – that is, an early token – arrives. It is meaningful only if the offered load is less than the maximum obtainable throughput of the ring, otherwise the interface queues continuously build up and the access delay gets progressively larger.

Providing the offered load is less than the maximum obtainable throughput, the maximum (worst case) access delay can also be deduced from Example 8.5. Recall that all stations simultaneously receive a set of frames at their ring interface queues such that they will each utilize the full TTRT quota for each rotation of the token. Assuming that station 1 is the first station able to transmit frames, the maximum access delay will be experienced by the last station on the ring – that is, station 4 – since this will receive a usable token only at the start of the fourth rotation of the token. Also, if th four frames in its ring interface queue are all addressed to station 3, then further delay equal to the ring latency will be experienced while they are al transmitted around the ring to this station. The general expression used compute the maximum access delay of an FDDI ring, A_{max}, is given by:

$$A_{max} = (n-1)\ (\text{TTRT} - T_l) + nT_l + T_l$$
$$= (n-1)\ \text{TTRT} + 2T_l$$

where A_{max} is the worst-case time to receive a usable token and the othe terms have the same meaning as before.

Clearly, since the ring latency is fixed for a particular ring configuratio the larger the ring TTRT, the larger the maximum access delay. As Exampl 8.6 shows, for all but the largest networks, the maximum obtainable throug put of an FDDI ring can be obtained with a TTRT equal to the minimu value of 4 ms. Hence although in the standard the maximum TTRT can be a high as 165 ms, it is common to operate the ring with a TTRT significant less than this.

le 8.6

Derive the maximum obtainable throughput and the maximum [illegible] delay for the following three ring configurations. Assume [illegible] 4 ms has been chosen for each configuration:

(i) 2 km ring with 20 stations

(ii) 20 km ring with 200 stations

(iii) 100 km ring with 500 stations

Answer:

The three ring configurations are the same as [illegible] and hence the same computed ring latencies [illegible]

Maximum available throughput η_{max} [illegible]

Maximum access delay A_{max} = $(n-1)$ TTRT [illegible]

(i) From Example 8.4, $T_l = 0.03$ ms. Hence

$$\eta_{max} = \frac{20(4-0.03)}{20\times4+0.03} = 0.99$$

and

$$A_{max} = 19\times4+0.06 = 76.06\text{ ms}$$

Continued

(ii) From Example 8.4, $T_l = 0.3$ ms. Hence

$$\eta_{max} = \frac{200(4-0.3)}{200\times4+0.3} = 0.92$$

and

$$A_{max} = 199\times4+0.6 = 796.6\text{ ms}$$

(iii) From Example 8.4, $T_l = 1$ ms. Hence

$$\eta_{max} = \frac{500(4-1)}{500\times4+1} = 0.75$$

and

$$A_{max} = 499\times4+2 = 1.998\text{ s}$$

8.7 High-speed LANs

As the application of LANs has become more diverse, so the demands on them in terms of information/data throughput have increased. As we have just described, by using a combination of bridges and a high bit rate backbone, the throughput of the total LAN is determined by the maximum throughput of each LAN segment. As we explained in Sections 8.3 and 8.4, the maximum throughput of the two basic LAN types is only a fraction of the bit rate that is used. Hence in order to meet the higher throughput requirements of the newer multimedia applications, a number of high-speed LAN types have been developed. These include variations of the basic Ethernet LAN and, since this is by far the most widely installed type of LAN, we shall restrict our discussion to three variations of Ethernet: (shared) **Fast Ethernet**, **Switched Fast Ethernet**, and **Gigabit Ethernet**.

8.7.1 Fast Ethernet

The aim of Fast Ethernet was to use the same shared, half-duplex transmission mode as Ethernet but to obtain a ×10 increase in operational bit rate over 10BaseT while at the same time retaining the same wiring systems, MAC method, and frame format. As we explained in Section 8.3, when using hubs with unshielded twisted-pair (UTP) cable, the maximum length of drop cable from the hub to a station is limited to 100 m by the driver/receiver electronics. Assuming just a single hub, this means that the maximum distance between any two stations is 200 m and the worst-case path length for collision detection purposes is 400 m plus the repeater delay in the hub. Clearly, therefore, a higher bit rate can be used while still retaining the sam CSMA/CD MAC method and minimum frame size of 512 bits. In the stan dard, the bit rate is set at 100 Mbps over existing UTP cable. Hence th standard is also known as **100BaseT**.

Line code

The major technological hurdle to overcome with Fast Ethernet was how t achieve a bit rate of 100 Mbps over 100 m of UTP cable. Category 3 UTP cab – as used for telephony, and the most widely installed – contains four separat twisted-pair wires. To reduce the bit rate used on each pair, all four pairs ar used to achieve the required bit rate of 100 Mbps in each direction.

With the CSMA/CD access control method, in the absence of contentio for the medium, all transmissions are half-duplex, that is, either station-to hub or hub-to-station. In a 10BaseT installation, just two of the four wire pai are used for data transfers, one in each direction. Collisions are detecte when the transmitting station (or hub) detects a signal on the receive pai while it is transmitting on the transmit pair. Since the collision detect func tion must also be performed in 100BaseT, the same two pairs are used for thi function. The remaining two pairs are operated in a bidirectional mode a shown in Figure 8.26(a).

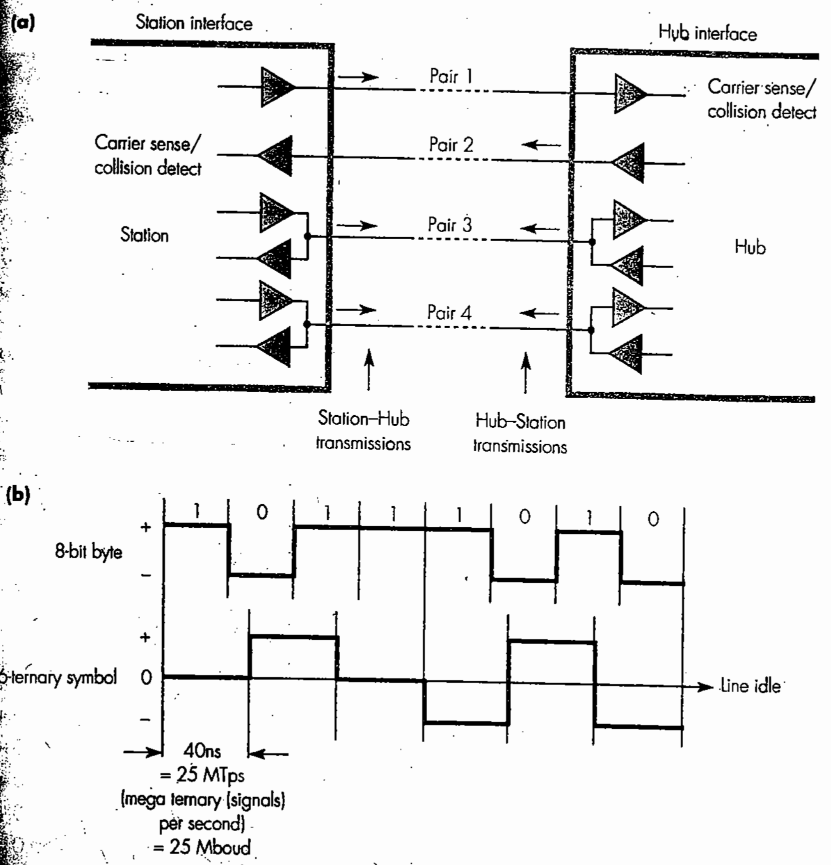
The figure shows that data transfers in each direction utilize three pairs pairs 1, 3, and 4 for transmissions between a station and the hub and pairs 2 3, and 4 for transmissions between the hub and a station. Transmissions o pairs 1 and 2 are then used for collision detection and carrier sense purpos as with 10BaseT. This means that the bit rate on each pair of wires need on be 100/3 = 33.33 Mbps.

If we use Manchester encoding, a bit rate of 33.33 Mbps requires a bau rate of 33.33 Mbaud which exceeds the 30Mbaud limit set for use with suc cables, as above this, unacceptably high levels of crosstalk are obtained reduce the baud rate, a 3-level (**ternary**) **code** is used instead of straigh level) binary coding. The code used is known as **8B6T** which means th prior to transmission, each set of 8 binary bits is first converted into 6 terna (3-level) symbols. From the example shown in Figure 8.26(b), we can dedu that this yields a symbol rate of:

$$\frac{100 \times 6/8}{3} = 25\,\text{Mbaud}$$

which is well within the set limit.

The three signal levels used are $+V$, 0, $-V$ which are represented simply +, 0, –. The codewords are selected such that the line is DC balanced, that the mean line signal is zero. This maximizes the receiver's discrimination o the three signal levels since these are then always relative to a constant 0 (DC level. To achieve this, we exploit the inherent redundancy present in the us of 6 ternary symbols. The 6 ternary symbols means that there are 729 (3


(a)
Station interface
Hub interface
Pair 1
Pair 2
Pair 3
Pair 4
Carrier sense/ collision detect
Carrier sense/ collision detect
Station
Hub
Station–Hub transmissions
Hub–Station transmissions
(b)
8-bit byte
1 0 1 1 1 0 1 0
+
–
6-ternary symbol
0
Line idle
40ns
= 25 MTps
(mega ternary (signals) per second)
= 25 Mbaud

8.26 100 Base T: (a) use of wire pairs; (b) 8B6T encoding.

possible codewords. Since only 256 codewords are required to represent the complete set of 8-bit byte combinations, the codes used are selected, firstly, to achieve DC balance and secondly, to ensure all codewords have at least two signal transitions within them. This is done so that the receiver DPLL maintains clock synchronization.

To satisfy the first condition, we choose only those codewords with a combined weight of 0 or +1 and 267 codes meet this condition. To satisfy the second condition, we eliminate those codes with fewer than two transitions – five codes – and also those starting or ending with four consecutive zeros – six codes. This leaves the required 256 codewords, the first 128 of which are listed in Table 8.2.

8.2 First 128 codewords of 8B6T codeword set.

a e	Codeword	Data byte	Codeword	Data byte	Codeword	Data byte	Codewor
	-+00-+	20	-++-00	40	-00+0+	60	[illegible]
	0-+-+0	21	+00+--	41	0-00++	61	[illegible]
	0-+0-+	22	-+0-++	42	0-0+0+	62	[illegible]
3	0-++0-	23	+-0-++	43	0-0++0	63	[illegible]
4	-+0+0-	24	+-0+00	44	-00++0	64	[illegible]
5	+0--+0	25	-+0+00	45	00-0++	65	[illegible]
6	+0-0-+	26	+00-00	46	00-+0+	66	[illegible]
7	+0-+0-	27	-+++--	47	00-++0	67	[illegible]
8	-+00+-	28	0++-0-	48	00+000	68	[illegible]
9	0-++-0	29	+0+0--	49	++-000	69	[illegible]
0A	0-+0+-	2A	+0+-0-	4A	+-+000	6A	[illegible]
0B	0-+-0+	2B	+0+--0	4B	-++000	6B	[illegible]
0C	-+0-0+	2C	0++--0	4C	0+-000	6C	[illegible]
0D	+0-+-0	2D	++00--	4D	+0-000	6D	[illegible]
0E	+0-0+-	2E	++0-0-	4E	0-+000	6E	[illegible]
0F	+0--0+	2F	++0--0	4F	-0+000	6F	[illegible]
10	0--+0+	30	+-00-+	50	+--+0+	70	[illegible]
11	-0-0++	31	0+--+0	51	-+-0++	71	[illegible]
12	-0-+0+	32	0+-0-+	52	-+-+0+	72	[illegible]
13	-0-++0	33	0+-+0-	53	-+-++0	73	[illegible]
14	[illegible]	34	+-0+0-	54	+--++0	74	[illegible]
15	[illegible]	35	[illegible]	55	[illegible]	75	[illegible]
16	[illegible]	36	[illegible]	56	[illegible]	76	[illegible]
17	[illegible]	37	[illegible]	57	[illegible]	77	[illegible]
18	[illegible]	38	[illegible]	58	[illegible]	78	[illegible]
19	[illegible]	39	[illegible]	59	[illegible]	79	[illegible]
1A	[illegible]	3A	[illegible]	5A	[illegible]	7A	[illegible]
1B	+00-+0	3B	0+--0+	5B	0+-0++	7B	[illegible]
1C	+00+-0	3C	+-0-0+	5C	+--0++	7C	[illegible]
1D	[illegible]	3D	-0++-0	5D	-000++	7D	[illegible]
1E	[illegible]	3E	-0+0+-	5E	0+++--	7E	[illegible]
1F	[illegible]	3F	-0+-0+	5F	0++-00	7F	[illegible]

DC balance

As we have just indicated, all the codewords selected have a combined weight of either 0 or +1. For example, the codeword + – – +00 has a combined weight of 0 while the codeword 0+++– – has a weight of +1. Clearly, if a string of codewords each of weight +1 is transmitted, then the mean signal level at the receiver will move away rapidly from the zero level, causing the signal to be misinterpreted. This is known as **DC wander** and is caused by the use of transformers at each end of the line. The presence of transformers means there is no path for direct current (DC).

To overcome this, whenever a string of codewords with a weight of +1 is to be sent, the symbols in alternate codewords are inverted prior to transmission. For example, if a string comprising the codeword 0 + + + – – is to be sent, then the actual codewords transmitted will be 0+ + + – – , 0 – – – + + , 0 + + + – – , 0 – – – + + , and so on, yielding a mean signal level of 0. At the receiver, the same rules are applied and the alternative codewords will be reinverted into their original form prior to decoding. The procedure used for transmission is shown in the state transition diagram in Figure 8.27(a).

To reduce the latency during the decoding process, the 6 ternary symbols corresponding to each encoded byte are transmitted on the appropriate three wire pairs in the sequence shown in Figure 8.27(b). This means that the sequence of symbols received on each pair can be decoded independently. Also, the frame can be processed immediately after the last symbol is received.

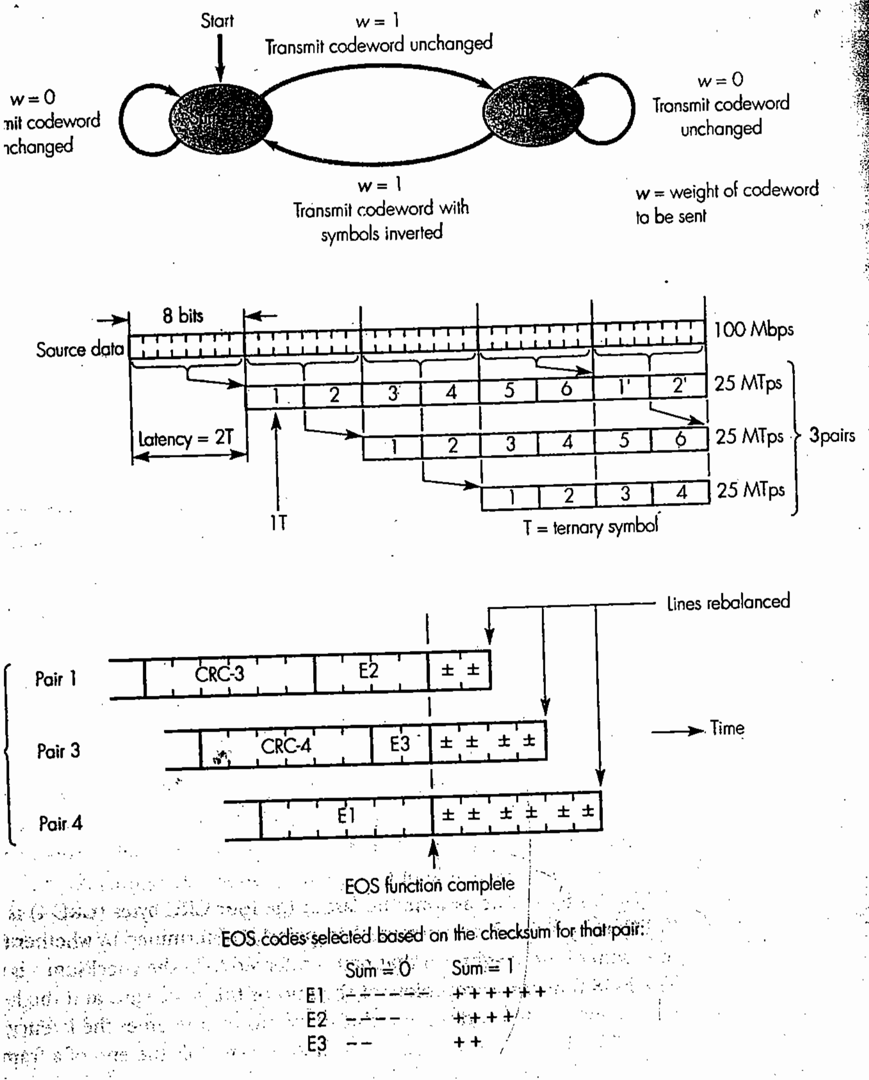
End-of-frame sequence

The transmission procedure adopted enables further error checking to be added to the basic CRC. We can deduce from the state transition diagram in Figure 8.27(a) that the running sum of the weights is always either 0 or +1. At the end of each frame transmission – that is, after the four CRC bytes have been transmitted – one of two different **end-of-stream (EOS)** codes is transmitted on each of the three pairs. The code selected effectively forms a checksum for that pair. The principle of the scheme is shown in Figure 8.27(c).

In this figure, we assume the last of the four CRC bytes (CRC-4) is on pair 3. The next codeword transmitted on pair 4 is determined by whether the running sum of the weights on that pair – referred to as the checksum – is 0 or +1. The EOS function is complete at the end of this codeword and the length of the other two EOS codes are reduced by two or one times the latency, that is, 4T or 2T. This means the receiver can detect reliably the end of a frame since all signals should cease within a short time of one another. This allows for very small variations in propagation delay on each pair of wires.

Collision detection

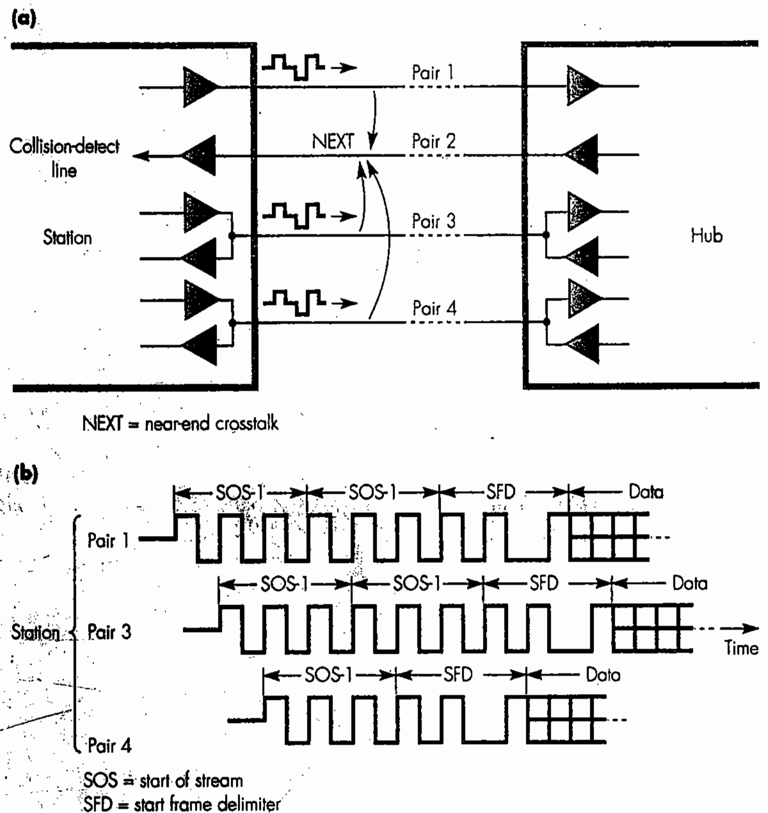
An example station–hub transmission without contention is shown in Figure 8.28(a). Recall that a station detects a collision by detecting a signal on pair 2 while it is transmitting and, similarly, the hub detects a collision by the


Start
w = 1
Transmit codeword unchanged
w = 0
nit codeword
nchanged
w = 0
Transmit codeword
unchanged
w = 1
Transmit codeword with
symbols inverted
w = weight of codeword
to be sent
8 bits
Source data
100 Mbps
1 2 3 4 5 6 1' 2'
25 MTps
1 2 3 4 5 6
25 MTps
3pairs
1 2 3 4
25 MTps
Latency = 2T
1T
T = ternary symbol
Lines rebalanced
Pair 1
CRC-3
E2
± ±
Pair 3
CRC-4
E3
± ± ± ±
Time
Pair 4
E1
± ± ± ± ± ±
EOS function complete
EOS codes selected based on the checksum for that pair:
Sum = 0
Sum = 1
E1 – – – – – –
+ + + + + +
E2 – – – –
+ + + +
E3 – –
+ +

8.27 100BaseT transmission detail: (a) DC balance transmission rules; (b) 8B6T ng sequence; (c) end of stream encoding.

presence of a signal on pair 1. However, as Figure 8.28(a) shows, the strong (unattenuated) signals transmitted on pairs 1, 3, and 4 from the station side each induce a signal into the collision detect – pair 2 – wire. This is near-end crosstalk (NEXT) and, in the limit, is interpreted by the station as a (collision) signal being received from the hub. The same applies for transmissions in the reverse direction from hub to station.

To minimize any uncertainty the preamble at the start of each frame is encoded as a string of 2-level (as opposed to 3-level) symbols, that is, only positive and negative signal levels are present in each encoded symbol. This increases the signal-level amplitude variations which, in turn, helps the station/hub to discriminate between an induced NEXT signal and the preamble of a colliding frame.


(a)
Pair 1
NEXT
Pair 2
Collision-detect
line
Pair 3
Station
Hub
Pair 4
NEXT = near-end crosstalk
(b)
SOS-1
SOS-1
SFD
Data
Pair 1
SOS-1
SOS-1
SFD
Data
Station
Pair 3
Time
SOS-1
SFD
Data
Pair 4
SOS = start of stream
SFD = start frame delimiter

igure 8.28 Start-of-frame detail: (a) effect of NEXT; (b) preamble sequence.

The preamble pattern on each pair is known as the **start of stream** (SO
and is made up of two 2-level codewords, SOS-1 and SFD. The complete pa
tern transmitted on each of the three pairs is shown in Figure 8.28(b) and a
we can see, the SFD codeword on each pair is staggered by sending only
single SOS-1 on pair 4. This means that the first byte of the frame is transm
ted on pair 4, the next on pair 1, the next on pair 3, and so on. An acceptab
start of frame requires all three SFD codes to be detected, and the staggerin
of them means that it takes at least four symbol errors to cause an und
tectable start-of-frame error.

On detecting a collision, a station transmits the jam sequence and the
stops transmitting. At this point, the station must be able to determine whe
the other station(s) involved in the collision cease transmitting in order
start the retry process. In practice, this is relatively easy since, in the nontrans
mitting (idle) state with 8B6T encoding, a zero signal level is present on the
three data wires. This means that there is no induced NEXT signal in the co
lision detect wire which, in turn, enables the completion of the jam sequence
from the hub side to be readily determined. Also, to improve the utilizatio
of the cable, the interframe gap time is reduced from 9.6 μs to 960 ns.

100BaseX

In addition to the 100BaseT standard, a second Fast Ethernet standard is ava
able, which is known as **100BaseX**. Unlike 100BaseT which was designed fo
use with existing category 3 UTP cable, 100BaseX was designed for use wi
the higher quality category 5 cable now being used in most new installatio
In addition, it is intended for use with STP and optical fiber cables. The use
various types of transmission media is the origin of the "X" in the name.

Each different type of transmission medium requires a different physi
sublayer. The first to be developed is that for use with multimode optical fib
cable as used in FDDI networks. Recall from the last section, the **FDDI LA**
intended primarily as a backbone subnetwork since, unlike 100BaseT, it
span a distance up to several kilometers. Transmissions over an **FDDI** netw
use a bit encoding scheme known as 4B5B (sometimes written as 4B/5B)
this has also been adopted for use with 100BaseX, this version being know
100BaseFX.

The cable comprises two fibers, one of which is used for transmissi
between the station and hub and the other for transmissions between
hub and the station. As with 10BaseT, collisions are detected if a (collidi
signal is present on the receive fiber during the time the station is trans
ting a frame. However, because of the additional cost of both
electrical-to-optical conversion circuits and the associated optical plugs a
sockets that are required per port, the cost of the MAC unit associated w
the NIC is higher than that used with 100BaseT. Hence the most popu
type of Fast Ethernet is 100BaseT and 100BaseFX is used primarily whe
longer drop cables are required.

8.7.2 Switched Fast Ethernet

As we explained at the start of Section 8.7.1, Fast Ethernet uses the same shared, half-duplex transmission mode as Ethernet. Hence in applications that involve access to, say, large enterprise Web servers, even though the server can handle multiple transfers concurrently, the overall access time and throughput experienced by the various stations using the server is limited by the shared access circuit connecting the server (station) to the hub.

In order to allow multiple access/transfers to be in progress concurrently, two developments have been made: the first, the introduction of a switched hub architecture, and the second, duplex working over the circuits that connect the stations to the hub. The resulting type of hub is known as a **Fast Ethernet switch.**

Switch architecture

The general architecture of a switching hub is shown in Figure 8.29. As we can see, each station is connected to the hub by means of a pair of (duplex) lines which, typically, are implemented as dual UTP (or STP) cables or dual multimode fiber cables. Recall from the last section, each UTP (and STP) cable contains four separate twisted pairs. In the case of 100BaseT, three pairs are used to transmit the 100 Mbps bitstream – in a half-duplex mode – and

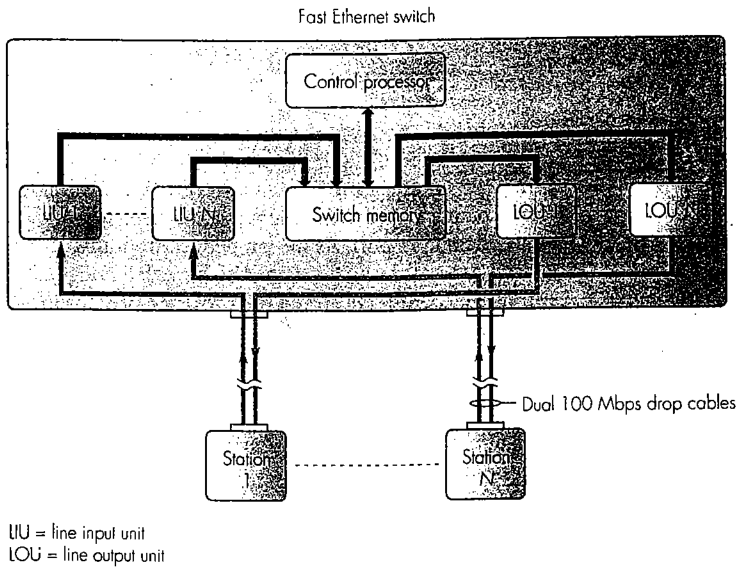


Figure 8.29 Fast Ethernet switch schematic.

the fourth pair is used to perform the carrier sense and collision detection functions. With a switching hub, however, CSMA/CD is not used and instead all stations can transmit and receive frames concurrently. Hence, as with 100BaseT, three pairs in each cable are used collectively to transmit frames (at 100 Mbps) in each direction.

In the case of dual multimode fiber cables, each fiber is used to transmit at 100 Mbps over several kilometers, one in each direction of transmission. Since the 4B5B coding scheme is used, the line signaling rate is 125 Mbaud. In addition, an active signal is maintained on each fiber continuously by transmitting an idle symbol during the idle period between frames. This ensures the receiver DPLL can maintain clock synchronism between successive frame transmissions.

Because each station can transmit frames simultaneously, a frame may be received at multiple input ports of the hub – and hence require processing – simultaneously. Similarly, two or more frames may require the same output line simultaneously. Hence associated with each input and output line is a memory buffer that can hold several frames waiting to be either processed (input) or transmitted (output). The frames – memory pointers to the start of the frame in practice – are stored in a FIFO queue. The control processor then reads the pointer to the frame at the head of each input queue in turn, obtains the destination MAC address from its head, and transfers the frame pointer to the tail of the required output queue to await transmission.

In order to retain the same connectionless mode of operation of the other LAN types, when the switch is first brought into service – and subsequently at periodic intervals – the control processor enters a learning state. This is similar to that used in transparent bridges which we described in Section 8.5.1. Hence when in the learning state, the switch simply initiates the onward transmission of a copy of each frame received from an input line onto all output lines. Prior to doing this, however, the control processor reads the source address from the head of the frame and keeps a record of this, together with the input port number on which the frame was received, in a routing table. The contents of the routing table are then subsequently used to route each received frame to a specific output port. As we can deduce from this, there is a store-and-forward delay associated with a switch. Also, as with a bridge, the FCS at the tail of each frame is used to check for the presence of transmission errors prior to the frame being forwarded and corrupted frames are discarded.

Flow control

As we can see from the above, under heavy load conditions it is possible for all the frame buffers within the switch to become full. At this point, therefore, the control processor must discard any new frame(s). Alternatively, an optional feature associated with switched hubs is to incorporate flow control into the switch. When using flow control, should the control processor find that the level of memory in use reaches a defined threshold, it initiates the transmission of what is called a **Pause** frame on all of its input ports.

On receipt of a Pause frame, the attached station must then stop sending any further frames to the switch until either a defined time has expired or it receives a notification from the switch that the overload condition has passed. Having sent a Pause frame, the control processor monitors the level of memory in use and, when this falls below a second level, it sends out a **Continue** frame on all input ports to inform the attached stations that they can now resume sending new frames.

Network configurations

An example (small) network configuration that includes a Fast Ethernet switching hub is shown in Figure 8.30. As we can see, in order to obtain a high level of throughput, the two servers are connected directly to the switch by means of duplex 100 Mbps lines. All the end-user stations then gain access to the servers through either a 10BaseT or a 100BaseT hub. As we explained in the last section, both these types of hub operate in the half-duplex repeating mode. Hence the duplex uplink port connecting each hub to the switch has bridging circuitry within the hub to temporarily buffer all frame transfers to and from the switch and to perform the CSMA/CD MAC protocol associated with the shared medium hub ports.

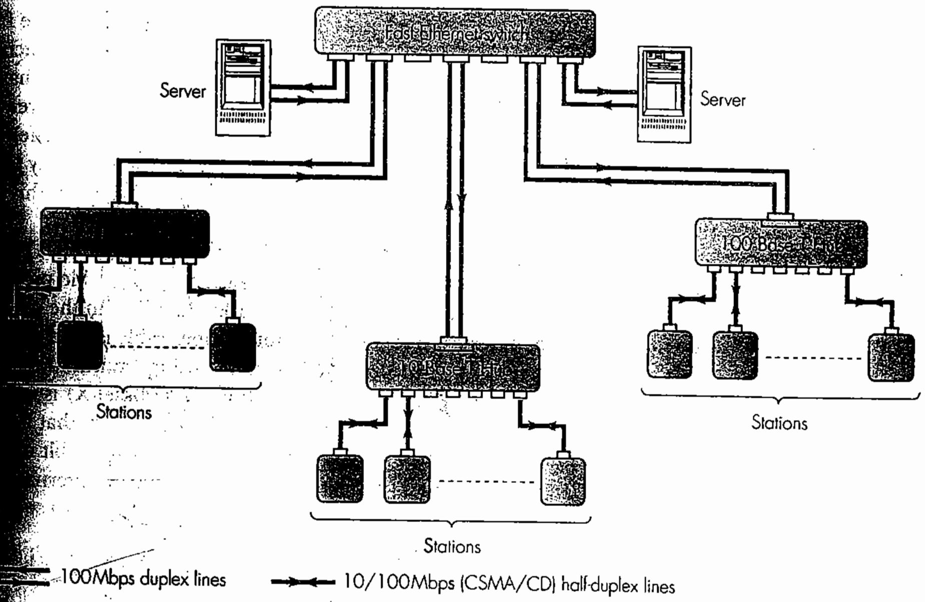


Figure 8.30 Example network configuration with a Fast Ethernet switch and 10/100BaseT hubs.

8.7.3 Gigabit Ethernet

As the name implies, the drop cables associated with Gigabit Etherne
operate at 1000 Mbps (1 Gbps). They have been introduced to mee
throughput demands of an increasing number of servers that hold files
taining multimedia information; examples include web pages compr
very high resolution graphics, motion video, and general audio. The hub
be either a simple repeater hub – that is, one that has no memory assoc
with it – or a switched hub.

An example application of a repeater hub is to distribute the outpu
powerful supercomputer (performing, say, 3D scientific visualizations)
localized set of workstations. The main issue when operating in the repe
mode is to ensure that the round-trip delay between any two stations –
slot time – exceeds the time required to transmit the smaller allowable fra
of 512 bits. The time to transmit a 512-bit frame at 1 Gbps is 0.512 micros
onds whereas the slot time is in the order of 2 microseconds. To overco
this, a technique known as **carrier extension** is used. This, as the na
implies, ensures that a known signal is present on the line for the duratio
the slot time and therefore a collision can still be detected. Alternative
technique called **frame bursting** can be used which allows multiple sh
frames to be transmitted one after the other up to the slot time.

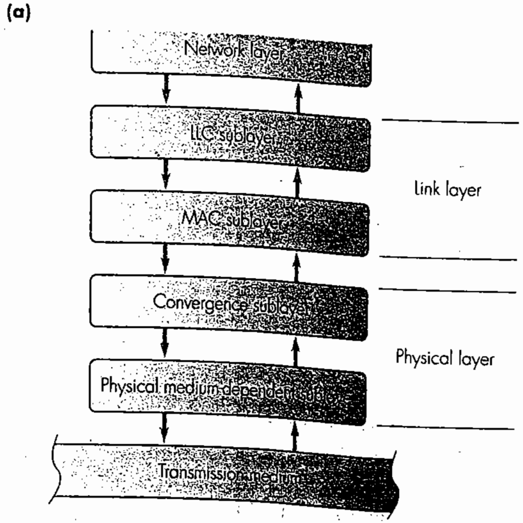
In the case of a switched hub, these are used to perform a similar rol
the Fast Ethernet switch we showed in Figure 8.30. As we can deduce fro
the figure, if all the attached hubs are 100BaseT – or Fast Ethernet switche
then the switch providing the backbone function must also be able to han
the increased load generated by the various hubs. Normally, however,
lines connecting each 100BaseT hub/switch to the Gigabit Ethernet
operate at 100 Mbps and only the servers that are connected directly to
Gigabit hub operate at the full speed of 1 Gbps. However, since in this
both hubs operate in the duplex mode, the slot time is not an issue.

In terms of cabling, repeating hubs can use either category 5 UTP
with a drop cable length of up to 100 m (1000BaseT) or STP cable prov
the length of the drop cable is limited to 25 m (1000BaseCX). In the cas
switched hub used as a backbone, however, optical fiber cable is used tha
ports drop cable lengths of up to 200 m – using multimode
(1000BaseSX) – or up to 10 km if monomode fiber (1000BaseLX) is use
the case of multimode fiber, an 8B/10B coding scheme is used; that is,
8-bit group is encoded into a 10-bit symbol. Hence, since the line bit rate
Gbps, the line signaling rate is 1.25 Gbaud. Apart from this, the internal a
tecture of the hubs is similar to those used in Fast Ethernet hubs and switch

8.8 LAN protocols

As we have learnt, there is a range of different types of LAN each of which uses a different MAC method, transmission mode, and transmission medium. In the context of our reference model, however, these differences only manifest themselves at the MAC sublayer (of the link layer) and the physical layer. The link control sublayer – known as the **logical link control (LLC) sublayer** in the context of LANs – then offers a standard link layer service to the network layer above it.

Apart from FDDI, which is defined in ISO9314, the various protocol standards associated with LANs are all part of the **IEEE 802 series**. The framework used for defining the various standards is shown in Figure 8.31 (a) and a selection of the protocols which we have described in the previous sections are listed in Figure 8.31 (b).



(b)

IEEE	802.1	Station management
	802.1d	Transparent bridges
	802.2	Logical link control (LLC)
	802.3	CSMA/CD (Ethernet) bus
	802.3u	Fast Ethernet
	802.3x	Hop-by-hop switch flow control
	802.3z	Gigabit Ethernet
	802.5	Token ring

Figure 8.31 LAN protocols: (a) protocol framework; (b) examples.

8.8.1 Physical layer

To cater for the different types of media and transmission bit rates, the phy cal layer has been divided into two sublayers: the **physical medium-dependen** **(PMD) sublayer** and the (physical) **convergence sublayer** (CS). To facilita the use of different media types, a **media-independent interface** (MMI) ha been defined for use between the convergence and PMD sublayers. The ro of the convergence sublayer is then to make the use of different media type and bit rates transparent to the MAC sublayer.

The set of signals associated with both interfaces vary slightly for the dif ferent types of Ethernet (802.3) and token ring (802.5). As an example, the set of signals associated with the different types of Ethernet are shown in Figure 8.32.

As we explained in Section 8.7.1, at bit rates in excess of 10 Mbps it is no possible to use clock encoding – for example Manchester – because the resulting high line signal transition (baud) rate would violate the limit set fo use over UTP cable. Instead, bit encoding and a DPLL are used and to

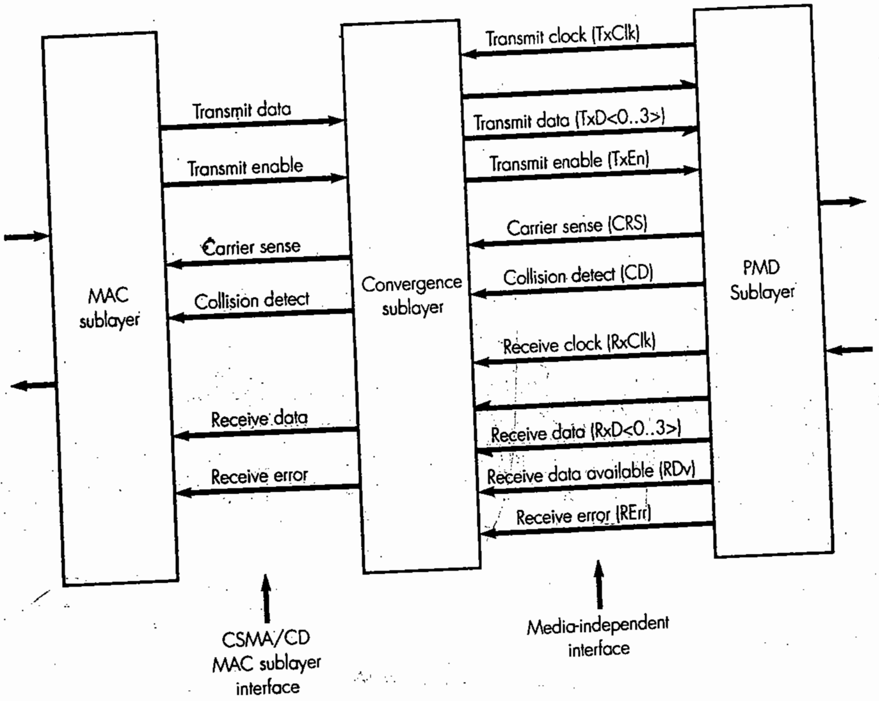


Figure 8.32 Fast Ethernet media-independent interface.

ensure the transmitted signal has sufficient transitions within it, the encoding schemes use one or more groups of 4 bits in each transmitted symbol. For example, one 4-bit group when 4B5B coding is used (100BaseSX), and two 4-bit groups when 8B/10B coding is used (1000BaseSX). Hence, all transfers over the MII are in 4-bit nibbles. The other control lines are concerned with the reliable transfer of these nibbles over the interface. The major functions of the convergence sublayer, therefore, are to convert the transmit and receive serial bitstream at the MAC sublayer interface into and from 4-bit nibbles for transfer across the MII, and, when half-duplex transmission is being used, to relay the carrier sense and collision detect signals generated by the PMD sublayer to the MAC sublayer.

8.8.2 MAC sublayer

Irrespective of the mode of operation of the MAC sublayer, a standard set of user service primitives is defined for use by the LLC sublayer. These are:

MA_UNITDATA.request
MA_UNITDATA.indication
MA_UNITDATA.confirm

A time sequence diagram illustrating their use is shown in Figure 8.33. For a CSMA/CD LAN, the confirm primitive indicates that the block of data associated with the request has been successfully (or not) transmitted – part (a) – while for a token ring LAN it indicates that the request has been successfully (or not) delivered, part (b).

Each service primitive has parameters associated with it. The MA_UNITDATA.request primitive includes: the required destination address (this may be an individual, group, or broadcast address), a service data unit (containing the data to be transferred – that is, the LLC PDU), and the required class of service associated with the PDU. This is used with token ring networks, for example, when a prioritized MAC protocol is being used.

The MA_UNITDATA.confirm primitive includes a parameter that specifies the success or failure of the associated MA_UNITDATA.request primitive. However, as we show in Figure 8.33, the confirm primitive is not generated as a result of a response from the remote LLC sublayer but rather by the local MAC entity. If the parameter indicates success, this simply shows that the MAC protocol entity (layer) was successful in transmitting the service data unit onto the network medium. If unsuccessful, the parameter indicates why the transmission attempt failed. As an example, if the network is a CSMA/CD bus, "excessive collisions" may be a typical failure parameter.

8.8.3 LLC sublayer

The LLC protocol is based on the high-level data link control (HDLC) protocol which we described earlier in Section 6.8. It therefore supports both a

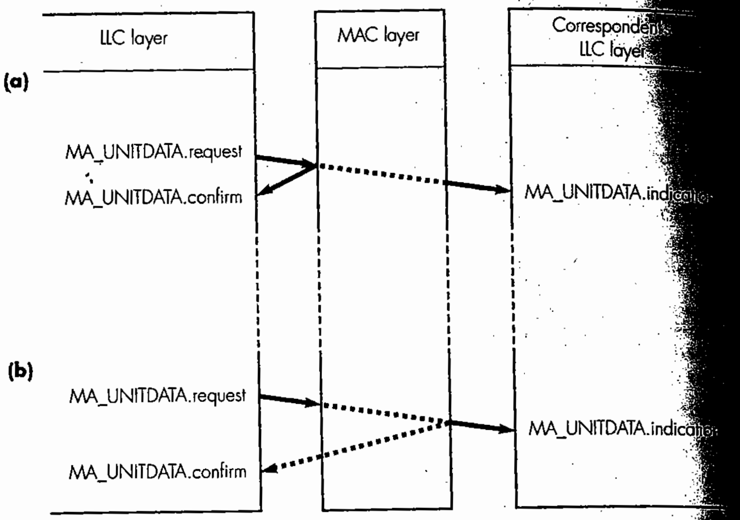

(a)
LLC layer
MAC layer
Corresponden
LLC layer
MA_UNITDATA.request
MA_UNITDATA.confirm
MA_UNITDATA.indicatio
(b)
MA_UNITDATA.request
MA_UNITDATA.confirm
MA_UNITDATA.indicatio

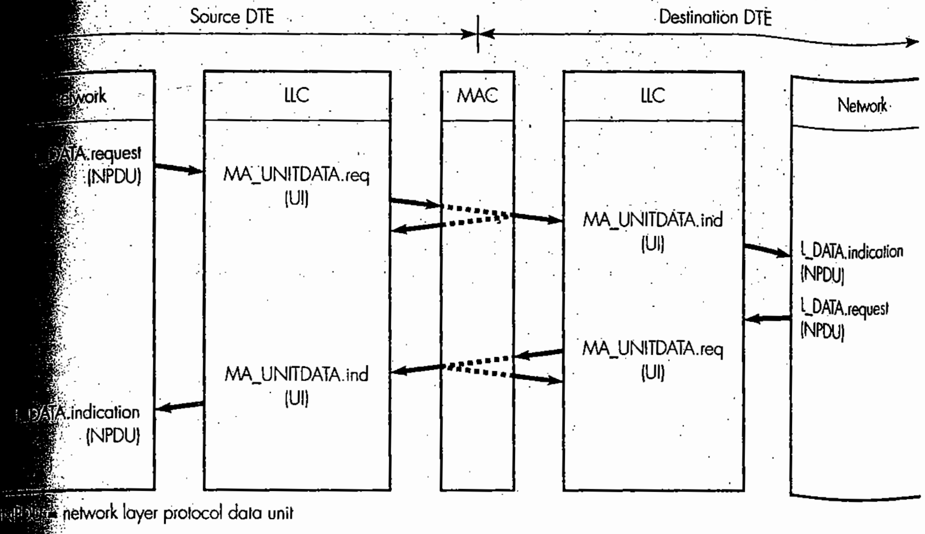
Figure 8.33 MAC user service primitives: (a) CSMA/CD; (b) token ring.

connectionless (best-effort) and a connection-oriented (reliable) mod almost all LAN networks, however, only the connectionless mode is us Hence, since this operational mode adds only minimal functionality, wh the older Ethernet MAC standard is being used – see the discussion on fr formats in section 8.3 – the LLC sublayer is often not present. Instead network layer – for example the Internet protocol (IP) – uses the ser provided by the MAC sublayer directly.

When the newer IEEE 802.3 MAC standard is being used, the LLC s layer is present. However, since it operates in the connectionless mode a only user service primitive is L_DATA.request and all data is transferred unnumbered information (UI) frame. The interactions between the LLC a MAC sublayers are as shown in Figure 8.34.

The L_DATA.request primitive has parameters associated with it. The are: a specification of the source (local) and destination (remote) addres and the user data (service data unit). The latter is the network layer prot data unit (NPDU). The source and destination addresses are each a conc nation of the MAC sublayer address of the station and an additional serv access point (SAP) address. In theory, this can be used for interlayer rout purposes within the protocol stack of the station. In applications such as th Internet, however, this feature is not used and both the destination S (DSAP) and the source SAP (SSAP) are set to AA (hex). In addition, two fu ther fields are added. Collectively, the two fields form what is called the subn

---

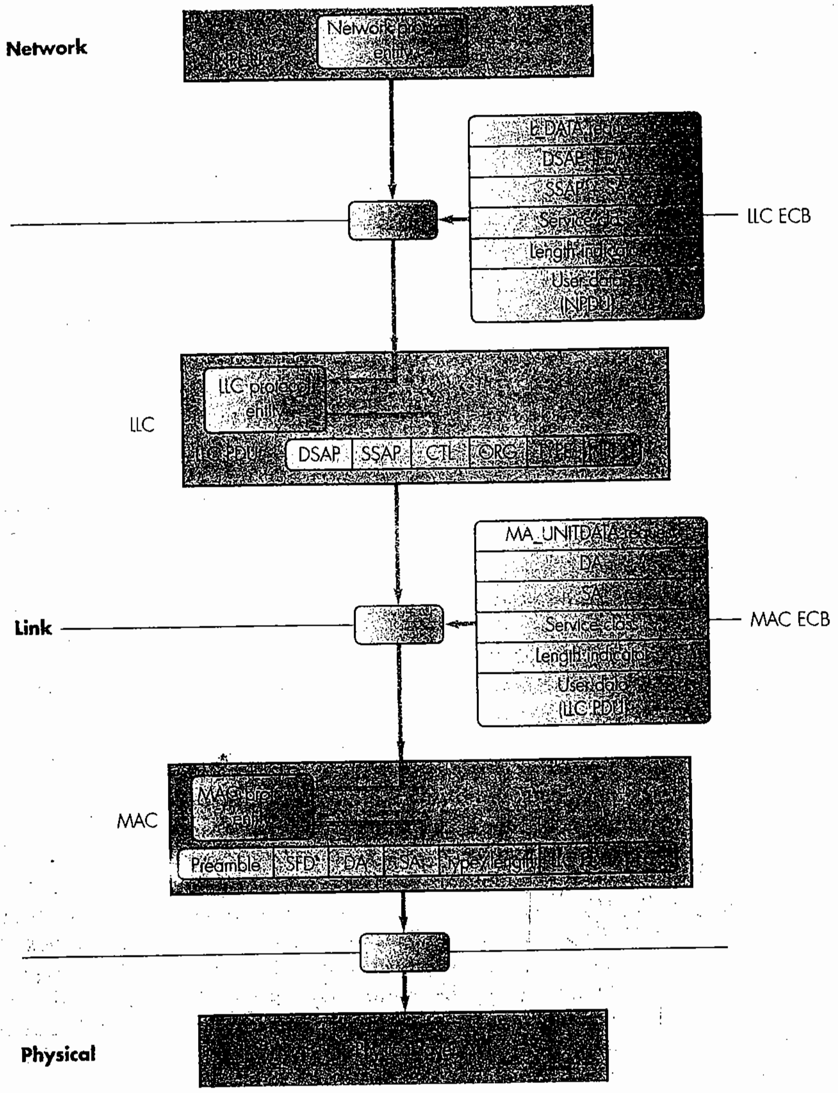

Source DTE
Destination DTE
twork
LLC
MAC
LLC
Network
DATA.request
(NPDU)
MA_UNITDATA.req
(UI)
MA_UNITDATA.ind
(UI)
L_DATA.indication
(NPDU)
L_DATA.request
(NPDU)
MA_UNITDATA.req
(UI)
MA_UNITDATA.ind
(UI)
DATA.indication
(NPDU)
network layer protocol data unit

ure 8.34 LLC/MAC sublayer interactions.

access protocol (**SNAP**) header. The first is a 3-byte field known as the *organisation* (*org*) *code* – which, with the Internet, all three bytes are set to zero – and a two-byte *type* field. This is the same as that used in the original Ethernet standard and indicates the network layer protocol that created the NPDU.

A more detailed illustration of the interactions between the LLC and MAC sublayers is shown in Figure 8.35. The LLC sublayer reads the destination and source LLC service access point addresses (DSAP and SSAP) – from the two address parameters in the event control block (ECB) associated with the L_DATA.request service primitive – and inserts these at the head of an LLC PDU. It then adds an 8-bit *control* (*CTL*) field – set to 03 (hex) to indicate it is an unnumbered information (UI) frame – the 3-byte org code, the type field, followed by the network layer protocol data unit in the user data field. The resulting LLC PDU is then passed to the MAC sublayer as the user data parameter of an MA_UNITDATA.request primitive in a MAC ECB. Other parameters include the MAC sublayer destination and source addresses (DA and SA), the desired service class, and the number of bytes (length indicator) in the user data field. Typically, the service class is used by the MAC sublayer protocol entity to determine the priority to be associated with the frame if a token network is being used.

On receipt of the request, the MAC protocol entity creates a frame ready for transmission on the link. In the case of a CSMA/CD bus network, it creates


Network
Network protocol entity
NPDU
L_DATA.request
DSAP + DA
SSAP + SA
Service class
Length indicator
User data (NPDU)
LLC ECB
LLC
LLC protocol entity
LLC PDU
DSAP
SSAP
CTL
ORG
TYPE
NPDU
MA_UNITDATA.request
DA
SA
Service class
Length indicator
User data (LLC PDU)
MAC ECB
Link
MAC
MAC protocol entity
Preamble
SFD
DA
SA
Type/length
LLC PDU
Physical
ECB = event control block

e 8.35 Interlayer primitives and parameters.

a frame containing the preamble and SFD fields, the DA and SA fields, an I-field, and the computed FCS field. The complete frame is then transmitted bit serially onto the cable medium using the appropriate MAC method.

A similar procedure is followed in the NIC of the destination station except that the corresponding fields in each PDU are read and interpreted by each layer. The user data field in each PDU is then passed up to the layer/sublayer immediately above together with the appropriate address parameters.

8.8.4 Network layer

The most popular protocol stack used within LANs is **Novell NetWare**. Hence it is common for all the stations connected to a site LAN to communicate using this stack. The network layer protocol associated with this is a connectionless protocol known as the **internet packet exchange (IPX)** protocol and, as its name implies, it can route and relay packets over the total LAN. There is no LLC sublayer associated with the stack and hence the IPX protocol communicates directly with the MAC sublayer.

The protocol stack used within the Internet is TCP/IP and the network layer protocol associated with this stack is a connectionless protocol known as the Internet protocol (IP). Hence, as we showed earlier in Figure 5.11, it is common for server machines such as email servers that need to communicate across the Internet, to support both protocol stacks, IPX to communicate with client stations over the site LAN and IP to communicate with other servers over the Internet. Since both IPX and IP are connectionless protocols, they use a single packet to transfer all information over the related network with the full network address of both the source and destination stations in the packet header. We shall defer further discussion of network layer protocols until Chapter 9 when we discuss the operation of the Internet.

8.9 Multisite LAN interconnection technologies

As we saw in the introduction to Section 8.2, in multisite enterprise networks, the LANs associated with the different sites are interconnected together to form an enterprisewide network. As we shall explain, various technologies are available to do this and the choice of technology is determined by the volume of intersite traffic. If this is low, then a low bit rate – up to 56 kbps – leased line with modems is used or, if this is not sufficient, one or more leased 64 kbps ISDN channels. Alternatively, if the volume of traffic is high, then high bit rate leased lines are used, typical bit rates being 1.5 or 2 Mbps or multiples of these. In addition, a number of high bit rate switched services have become available from some telecom providers.

The aim of an enterprise network is to provide an integrated structure so that all communications across the total enterprise ARE transparent to the site/location of the employee's PCs/workstations and the various enterprisewide servers. Hence providing a physical link between sites is only the first step in connecting the LANs at the various sites together. Associated each LAN is a gateway that provides the additional higher-level protocol port needed to ensure that the physical location of the application services is transparent to the end users.

In this section, we first describe the operation of the typical intersite ga ways in Section 8.9.1. The first is based on remote bridges and the second IP/IPX routers. We then describe two examples of switched services: switche ISDN (in Section 8.9.2) and frame relay in Section 8.9.3. A typical enterpris network architecture based on high bit rate leased circuits is then describe in Section 8.9.4.

8.9.1 Intersite gateways

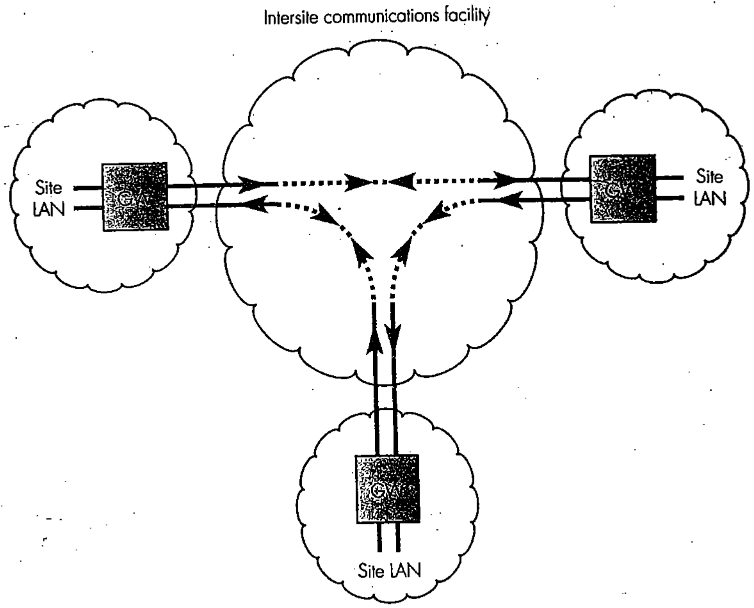
The simplest way of providing a transparent connection between sites is use a **remote bridge** as the intersite gateway. Normally, such bridges are co nected directly to the LAN backbone on one port and to the intersi line/network termination on the other. A similar bridge is then used at each site. An example network architecture is shown in Figure 8.36.

As we explained in Section 8.3, the 48-bit MAC address used in LANs route frames are built into the MAC chipset at the time of its manufactu and each contains a unique address. In practice, therefore, it is possible treat the interconnected set of site LANs as a single LAN and use just th MAC addresses at the head of each frame for routing.

As we have just explained, various schemes are used to set up connectio between sites. Irrespective of the scheme used, however, there is a separa connection set up between each site and hence each pair of LANs. Thus eve though each remote bridge has only two physical ports, logically, they beha like multiport bridges with a separate link to each of the other remote bridge Normally, the spanning tree algorithm we described in Section 8.5.1 spa only a single site/LAN and the remote bridge at the site does not take par the derivation of the active spanning tree. It does, however, perform the bas learning and forwarding/filtering functions. Hence after the learning pha it can route all subsequent frames over their appropriate intersite link.

As we show in Figure 8.36, an alternative (and more flexible) way of p viding a transparent connection between sites is to use an IP/IPX route the gateway. Routers are used when either the site LANs are of different ty and/or the intersite communications facility is itself a network and requi routing. In both these cases, MAC addresses cannot be used for routing p poses as they only have local significance to a particular site. Hence routi must be carried out at the network layer using either IP or IPX.

As we explained earlier in Section 8.8.4, both IP (as used with th Internet) and IPX (as used with Novell NetWare) are connectionless protocols and all information is transferred over the intersite communication facility in self-contained packets. At the head of each packet is the (enterprisewide) network address of both the source station and the destination station. As we shall expand upon in the next chapter when we discuss the operation of the Internet, each network address comprises a network identifier, which identifies uniquely each site network (LAN) within the context of the total enterprise, and a station identifier which identifies each station at that site. All routing over the intersite network is then carried out using these network addresses. It is for this reason that, when the IP protocol is used, the enterprise network is known also as an intranet. We shall defer further discussion of how the routing is carried out until Section 9.6 when we describe the operation of the IP protocol in some detail.



GW = intersite gateway (remote bridge or IP/IPX router)

Figure 8.36 Example enterprise network architecture.

8.9.2 ISDN switched connections

As we explained in Section 7.2.3, a basic rate interface to an ISDN offers two, separately-switched 64 kbps digital channels. Also, with an ISDN, the connection setup time of a channel is only a fraction of a second. Hence for small enterprises, one solution is to set up (and close down) a channel on demand. For medium-sized enterprises, although the same approach is often acceptable, multiple 64 kbps channels are required to meet the intersite bandwidth requirements. As we explained in Section 7.2.3, to meet this type of requirement a service known as **ISDN multirate** is available from some telecom providers that allows the user to request the bandwidth of a call to be any multiple of 64 kbps. The service is also known as switched $p \times 64$ where p can be up to either 23 or 30 depending on the provider. In order to use the multiple channels as a single high bit rate channel, however, it is necessary to use a device called an **inverse multiplexer**. The general scheme is shown in Figure 8.37(a).

We introduced the concept of multiplexing in Section 7.2.3 as a means of transmitting multiple low bit rate – 64 kbps – digitized voice signals over a single high bit rate line. In contrast, inverse multiplexing is used to derive a single high bit rate channel using multiple lower bit rate channels. For example, an inverse multiplexer can be used to derive a single 384 kbps channel from six independent 64 kbps channels.

At the sending side, the inverse multiplexer first sets up the appropriate number of 64 kbps channels to the required destination. It then proceeds to segment the high bit rate digital stream output by the user equipment – for example a remote bridge – ready for transmission over the multiple low bit rate channels. At the receiving side, once the multiple channels have been set up, the inverse multiplexer accepts the bitstreams received from these channels and reassembles them back into a single high bit rate channel for onward transmission to the receiving terminal equipment.

Hence the function of the inverse multiplexers, in addition to call setup, is to make the segmentation and reassembly operations transparent to the user equipment. In practice, because each channel is set up independently, they may all traverse the ISDN trunk network across different routes/paths. This results in small time differences in the signals received from each channel. To compensate for this, the inverse multiplexer at the receiving side must perform delay compensation and resynchronization of the reassembled bitstream. The general scheme is shown in Figure 8.37(b).

To obtain a similar service to that provided by switched $p \times 64$, inverse multiplexers are avaliable that enable the user terminal equipment to set up and clear multiple 64 kbps circuits on demand. The technique is known as **bonding** and the principles of the scheme are shown in Figure 8.37(c).

In order to set up the high bit rate channel – in response to a user request – the inverse multiplexer at the calling side requests a single 64kbps channel to be set up to the remote site. Once this is in place, the inverse multiplexer at the called side sends back a list of available local (extension) numbers. The calling multiplexer sets up the required number of additional channels – one at a time – and the two user equipments can then exchange data. During the call, a user terminal equipment can request that the aggregated bandwidth available is either increased or decreased by setting up or clearing channels dynamically. For example, if the terminal equipments are remote bridges then the aggregate bandwidth can be regulated to match the data traffic being exchanged at any point in time. An international standard has now been defined in **ITU-T recommendation H.221**, which is concerned with the operation of this type of equipment.

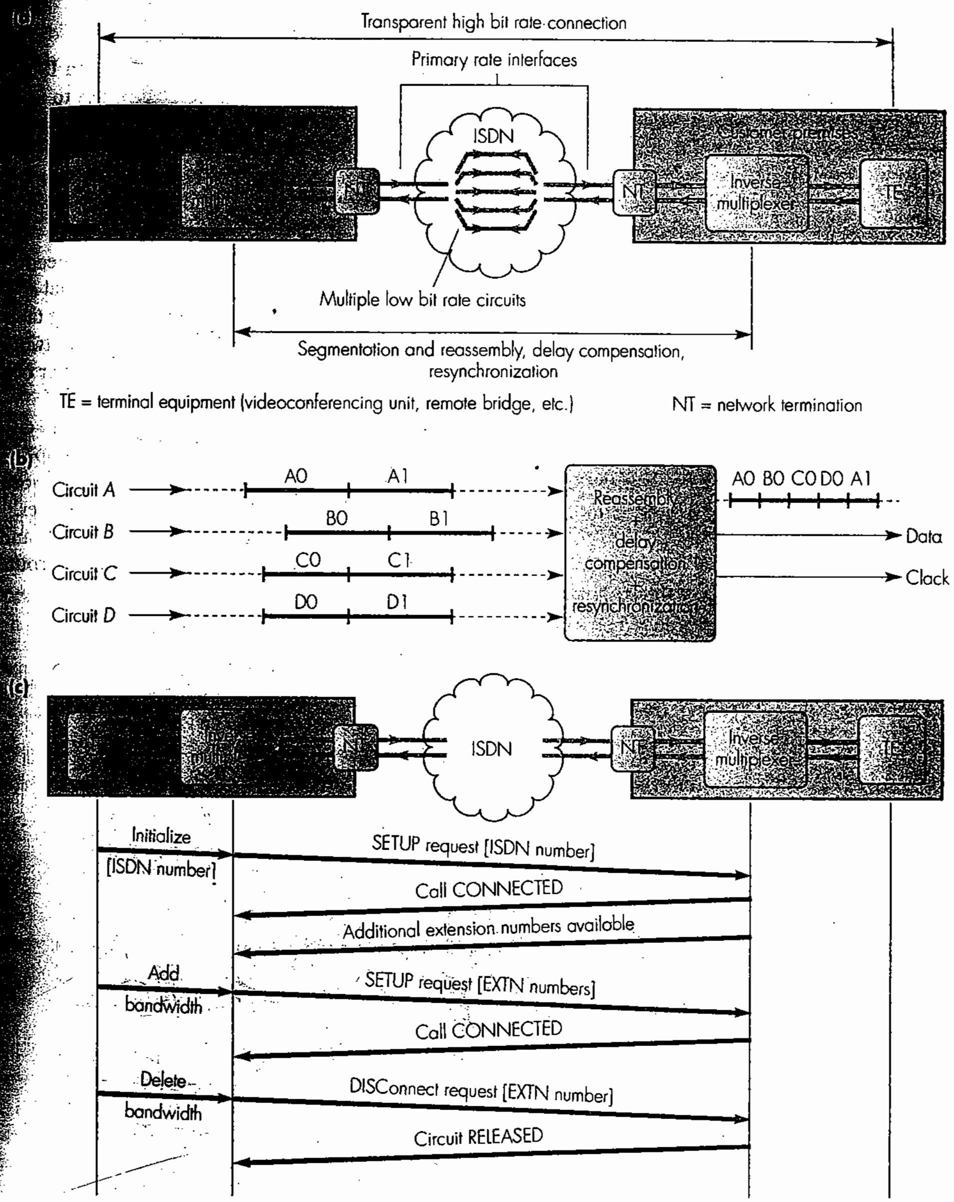

(a)
Transparent high bit rate connection
Primary rate interfaces
ISDN
NT
NT
Customer premises
Inverse multiplexer
TE
Multiple low bit rate circuits
Segmentation and reassembly, delay compensation, resynchronization
TE = terminal equipment (videoconferencing unit, remote bridge, etc.)
NT = network termination
(b)
Circuit A
Circuit B
Circuit C
Circuit D
A0
A1
B0
B1
C0
C1
D0
D1
Reassembly, delay compensation, resynchronization
A0 B0 C0 D0 A1
Data
Clock
(c)
NT
ISDN
NT
Inverse multiplexer
TE
Initialize [ISDN number]
SETUP request [ISDN number]
Call CONNECTED
Additional extension numbers available
Add bandwidth
SETUP request [EXTN numbers]
Call CONNECTED
Delete bandwidth
DISConnect request [EXTN number]
Circuit RELEASED

Figure 8.37 Inverse multiplexing: (a) principle of operation; (b) reassembly schematic; (c) bonding protocol.

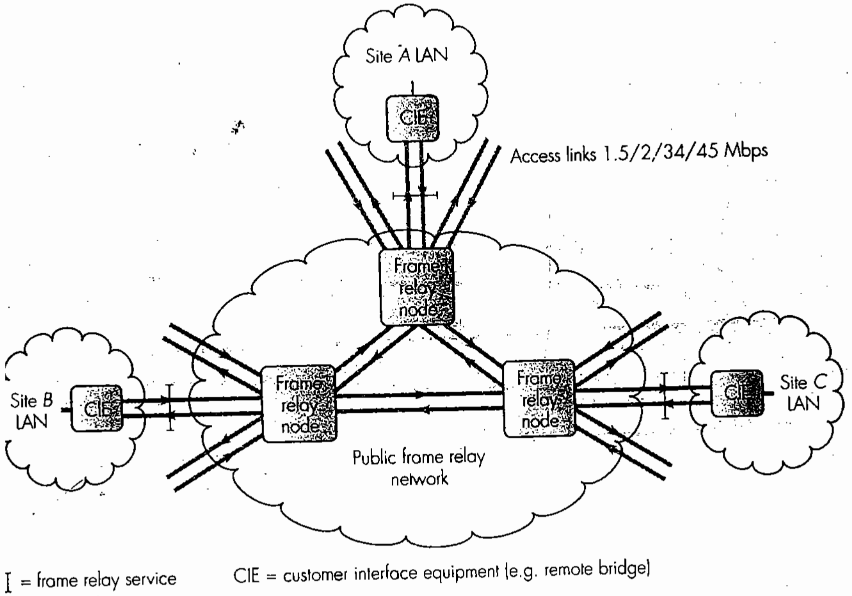
8.9.3. Frame relay

Frame relay was initially defined as a service provided through an Subsequently a number of telecom providers have set up networks to just a frame relay service. As the name implies, with frame relay, the plexing and routing of frames is performed at the link layer. Moreove routing of frames is very straightforward so the channel bit rate can be typical rates being up to 34 or 45 Mbps. A schematic diagram of a typ public frame relay network is shown in Figure 8.38.

The customer first informs the service provider of the sites that nee be interconnected. The provider then creates a set of permanent virtual nections that interconnect all the sites by setting up appropriate routing entries in each frame relay node. The provider then informs the netwo manager at each site of the identifier that should be put into the header frame to reach each of the other sites.

All frames are multiplexed together onto the link connecting the cu **tomer interface equipment** (**CIE**) to its nearest node. Logically, this appea to the customer equipment like a set of point-to-point connections betwee itself and all the other sites, each identified by the corresponding identifier

The identifier is known as the **data link connection identifier** (**DLCI**) an is put into the header of each frame. It is then used by the network to rou (relay) the frame to its intended destination.


Site A LAN
CIE
Access links 1.5/2/34/45 Mbps
Frame relay node
Site B LAN
CIE
Frame relay node
Frame relay node
Site C LAN
CIE
Public frame relay network
I = frame relay service
CIE = customer interface equipment (e.g. remote bridge)

8.38 Public frame-relay network schematic.

The format of each frame is based on that used in the HDLC protocol and is shown in Figure 8.39(a). It comprises a 2-byte (extended) address header with no control field owing to the lack of any error control. In addition to the DLCI, the header contains the **forward explicit congestion notification (FECN) bit**, the **backward explicit congestion notification (BECN) bit** and the **discard eligibility (DE) bit**. These are used for controlling congestion within the network.

The DLCI, like the VCI in packet-switched networks, has only local significance on a specific network link and therefore changes as a frame traverses the links associated with a virtual path. When the virtual path is first set up, an entry is made in the routing table of each frame relay node along the route of the incoming link/DLCI and the corresponding outgoing link/DLCI to reach the intended destination of the frame. An example set of entries is shown in Figure 8.39(b) and the related routing of each frame is illustrated in Figure 8.39(c).

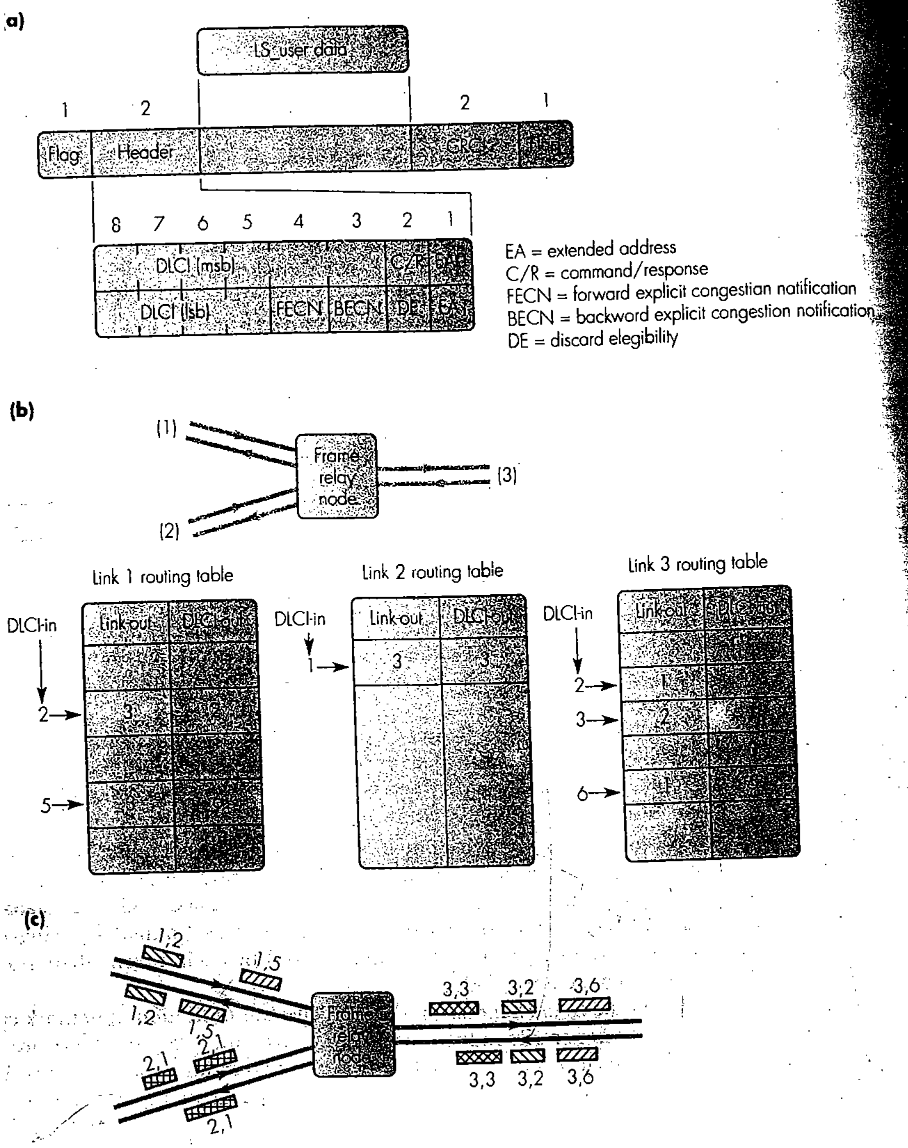
When a frame is received, the frame handler within each node simply reads the DLCI from within the frame and combines this with the incoming link number to determine the corresponding outgoing link and DLCI. The new DLCI is written into the frame header and the frame is queued for forwarding on the appropriate link. The order of relayed frames is thus preserved and their routing is very fast.

Since multiple frame transfers can be in progress concurrently over each link within the network, during periods of heavy traffic an outgoing link may become temporarily overloaded resulting in its queue starting to build up. This is known as **congestion** and the additional congestion control bits in each frame can be used to alleviate this condition.

Whenever the frame handler relays a frame to a link output queue, it checks the size of the queue. If this exceeds a defined limit, the frame handler signals this condition to the two end users involved in the transfer. This is done in the forward direction by setting the FECN bit in the frame header. In the backward direction, it is done by setting the BECN bit in the header of all frames which are received on this link. If the condition persists, the frame handler also returns a special frame called a **consolidated link layer management** (**CLLM**) frame to all CIEs that have routes (paths) involving the affected link. Such frames are simply relayed by each intermediate frame relay node in the normal way.

When the frame handler in a CIE receives an indication of network congestion, it temporarily reduces its frame forwarding rate until there are no further indications of congestion. If the overload increases, however, the frame relay node must start to discard frames. In an attempt to achieve fairness, the DE bit in the frame header is used since this is set by the frame handler in each CIE whenever the rate of entry of frames into the network exceeds the peak rate agreed at subscription time.

To minimize the possibility of wrongly delivered frames, the CRC at the tail of each frame is used to detect bit errors in the frame header (and information) fields. Then, if an error is detected, the frame is discarded. With the


(a)
LS user data
1
2
2
1
Flag
Header
CRC
8 7 6 5 4 3 2 1
DLCI (msb)
C/R
EA
DLCI (lsb)
FECN
BECN
DE
EA
EA = extended address
C/R = command/response
FECN = forward explicit congestion notification
BECN = backword explicit congestion notification
DE = discard elegibility
(b)
(1)
(2)
(3)
Frame relay node
Link 1 routing table
Link 2 routing table
Link 3 routing table
DLCI-in
Link-out
DLCI-out
2
5
1
3
3
2
3
6
(c)
1,2
1,5
2,1
3,3
3,2
3,6
Frame relay node

ure 8.39 Frame relay principles: (a) frame format; (b) frame routing; (c) frame relay hematic.

frame relay service, error recovery is left to the higher protocol layers in the end-user stations. The cost of the service is based on the actual number of frames transferred.

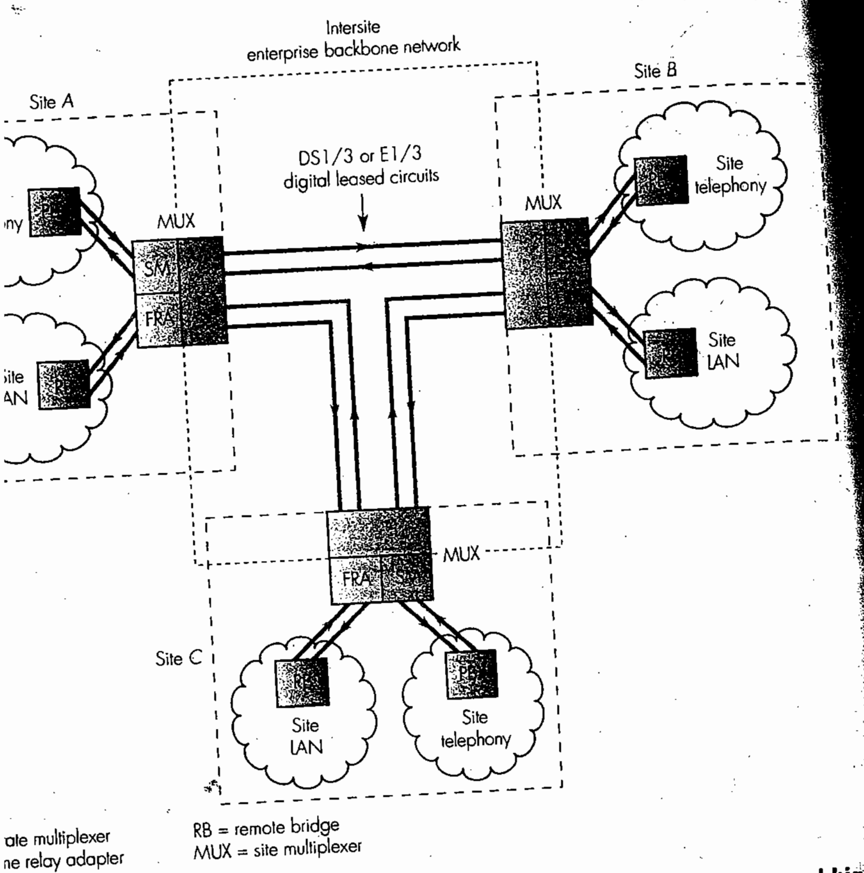
8.9.4 High bit rate leased lines

A typical large multisite enterprise network based on high bit rate leased lines is illustrated in Figure 8.40. As we explained in Section 8.1, in addition to LAN (data) traffic, the network must also support intersite telephony traffic. Typically, the leased lines are either DS1/T1 (1.5 Mbps) or E1 (2 Mbps) or higher such as DS3 (44 Mbps) and E3 (34 Mbps). The multiple 64 kbps channels these contain are divided between telephony – for PBX interconnection – and data – for LAN interconnection – on a semipermanent basis using network management within each site multiplexer.

In the case of telephony, although a 64 kbps channel is used by the PBX for each call, it is now common to use each 64 kbps channel of the intersite leased circuit for more than one call. As we explained earlier in Section 4.2, there is now a range of compression algorithms/circuits available that provide good quality voice communication using 32, 16, or 8 kbps. This means that 2, 4, or 8 calls can be multiplexed into a single 64 kbps channel giving a substantial saving in the number of channels required between sites. The technique used to do this is known as **subrate multiplexing** and is particularly worthwhile over costly international leased lines.

In the case of data traffic, a common approach is to use a (private) **frame relay adapter (FRA)**. These operate in a similar way to the frame relay nodes in a public frame relay network. The traffic between each pair of sites is allocated a portion of the bandwidth – number of 64 kbps channels – of the related leased line by network management. A set of DLCIs are then assigned to each path/route and these are loaded by network management into the routing tables of the interconnected set of FRAs. A related DLCI for each path is also passed to the remote bridge (or router) at each site and, once the set of MAC addresses associated with each path have been learnt by the bridge, it writes the appropriate DLCI in the header of each frame prior to passing the frame to its local FRA. The role of the FRA is then to relay each frame received from the remote bridge using the preassigned entries in its routing table and the corresponding set of aggregated channels.

Although not shown, there is a single network management station for the total network and all the devices shown in the figure have network management (agent) software within them. Normally, the management station is connected to one of the site multiplexers and a single 64 kbps channel of the intersite leased circuits is then used to transfer management-related messages to/from the management station and all the other network devices. The messages include configuration information to all devices – routing table entries, bandwidth allocations, and so on – and fault reports from the devices. In this way, should a fault develop or a reconfiguration be necessary, this can be done from a central site in a secure way.

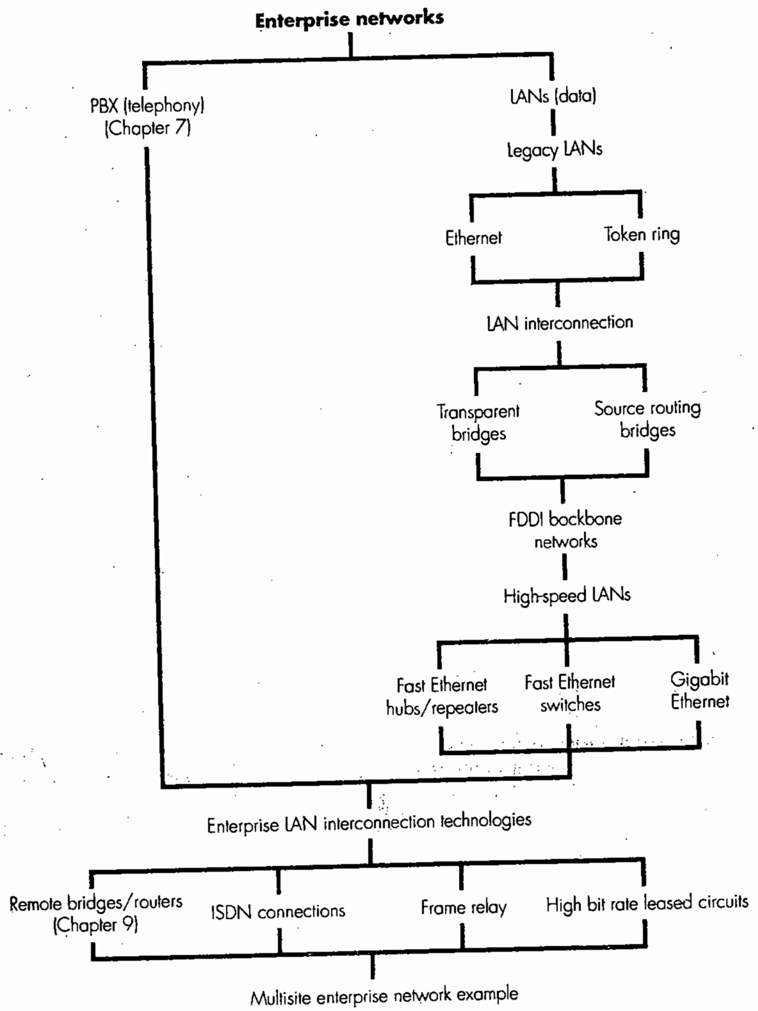

Intersite
enterprise backbone network
Site A
Site B
DS1/3 or E1/3
digital leased circuits
MUX
SM
FRA
ny
ite
AN
MUX
Site
telephony
Site
LAN
MUX
FRA
Site C
Site
LAN
Site
telephony
ate multiplexer
ne relay adapter
RB = remote bridge
MUX = site multiplexer

40 Schematic of large multisite enterprise network based on multiplexers and hig
eased circuits.

Finally, although most private networks are run and managed by th
enterprise to which they belong, a number of telecom providers now off
the option for an equivalent private network to be set up within th
provider's network. This is then known as a **virtual private network (VPN**
These offer the same set of services to the private network but are manage
and operated by the telecom provider. In general, VPNs are more expensiv
than private networks but they have the advantage that the enterprise nee
not then be involved in the recruitment of staff who are not concerned wit
the core business of the enterprise.

Summary

A summary of the various topics discussed in this chapter is given in Figure 8.41.


Enterprise networks
PBX (telephony)
(Chapter 7)
LANs (data)
Legacy LANs
Ethernet
Token ring
LAN interconnection
Transparent
bridges
Source routing
bridges
FDDI bockbone
networks
High-speed LANs
Fast Ethernet
hubs/repeaters
Fast Ethernet
switches
Gigabit
Ethernet
Enterprise LAN interconnection technologies
Remote bridges/routers
(Chapter 9)
ISDN connections
Frame relay
High bit rate leased circuits
Multisite enterprise network example

igure 8.41 Summary of the topics discussed in this chapter relating to enterprise networks.

Exercises

tion 8.1

What is the meaning of the term "enterprise network"? Describe the factors that determine when such networks are created.

ction 8.3

Explain the meaning of the following terms relating to the CSMA/CD medium access control method:
(i) multiple access,
(ii) broadcast mode,
(iii) collision,
(iv) carrier sense.

3 With the aid of Figure 8.1, explain the meaning of the term "slot time" and how this is derived. State how the slot time determines the maximum throughput of the LAN.

4 State the use of a jam sequence with the CSMA/CD MAC method and explain why a truncated binary exponential process is used.

.5 Explain the origin of the hub configuration as is now used for Ethernet LANs. Also, with the aid of a diagram, explain how the broadcast mode of operation is achieved.

.6 Two UTP hubs to which user stations are attached are each connected to a third hub by optical fiber cable in order to gain access to a server that is attached to the third hub. Derive the maximum length of optical fiber cable that can be used.

8.7 With the aid of the frame format shown in Figure 8.3(a), explain:
(i) the clock encoding method used and how the start of a new frame is detected,
(ii) how each station that receives a frame determines from the destination address whether the frame contents are intended for it,
(iii) the use of the type/length field.

8.8 Explain the meaning and use of:
(i) an interface gap,
(ii) a backoff limit,
(iii) an attempt limit.
Hence, with the aid of flow diagrams, describe the transmit and receive procedures followed by the CSMA/CD MAC sublayer.

Section 8.4

8.9 With the aid of Figure 8.5, explain how the tran mission of frames over the ring is controlle using a token. Include in your explanation th two alternatives that are used by a station to release the token.

8.10 With the aid of the two wiring configuration used with a token ring shown in Figure 8.6 explain the meaning/use of the following:
(i) hub/concentrator,
(ii) station coupling unit,
(iii) trunk coupling unit.

8.11 Explain the meaning/use of the followin relating to a token ring:
(i) active ring monitor,
(ii) minimum ring latency,
(iii) elastic buffer.

8.12 With the aid of the diagrams in Figure 8 explain:
(i) how the start and end of a frame detected,
(ii) the role of the token, monitor, prio and reservation bits in the access cont field,
(iii) the role of the A and C bits in the fra status field.

8.13 With the aid of flow diagrams, explain t transmission and reception procedures o frame with a token ring LAN. Include t meaning/use of the token hold timer.

8.14 Assume the same operating conditions as us in Example 8.1. After a period of inactivity, tions 1, 2, and 3 generates a frame to send the priority of the frames is 2, 0, and 2 respectively, derive and show in table form the transmissions made by each station for five further rotations of the token.

Section 8.5

8.15 How is a bridge different from a repeater? What are the advantages and disadvantages of each?

8.16 With the aid of the bridge architecture shown in Figure 8.12(a), explain the operation of a bridge including the meaning of the terms:
(i) promiscuous mode,
(ii) forwarding database,
(iii) bridge learning.

8.17 With the aid of Figure 8.12(b), explain how the entries in the two forwarding databases would be built up assuming the following message exchanges: 1 to/from 3, 1 to/from 5, 2 to/from 4, 2 to/from 6.

8.18 With the aid of the LAN topology shown in Figure 8.13, explain why the learning process described in your solution to Exercise 8.16 would not work. Hence state how the topology needs to be changed.

In relation to the spanning tree algorithm, explain the meaning of the terms:
(i) spanning tree,
(ii) root bridge,
(iii) designated cost,
root path cost,
root port,
designated port.

ssume the same bridged LAN topology as own in Figure 8.14. Determine the active panning tree) topology for the following onditions:
(i) bridge B1 fails
(ii) all bridges are in service but segments S1, 2, 4, and 5 have three times the designated cost of segments S3 and 6 (that is, they have a higher bit rate)
(iii) the same designated costs as in (ii) but bridge B5 has a higher priority than the other bridges.

8.21 With the aid of the LAN topology shown in Figure 8.15, state the routing entry held by station A to reach station D. Hence describe how a frame sent by A is routed to D. Include in your description the structure of the routing information carried in the frame header.

8.22 In relation to the route discovery algorithm used with a source routing bridged LAN, explain the meaning/use of
(i) a single-route broadcast frame,
(ii) an all-routes broadcast frame.
Hence, with the aid of the LAN topology shown in Figure 8.17, explain how the indicated designation of each port is derived.

Section 8.6

8.23 With the aid of the LAN topology shown in Figure 8.18, explain the meaning/use of:
(i) multiport bridge,
(ii) a building backbone,
(iii) a site-wide backbone.

8.24 With the aid of Figure 8.19, explain the meaning/use of the following relating to an FDDI backbone:
(i) primary and secondary rings,
(ii) single attach and dual attach stations,
(iii) optical coupling unit,
(iv) patch panel wiring concentrator,
(v) polarized duplex connectors.

8.25 State why 4B/5B encoding is used in FDDI LANs rather than differential Manchester encoding.

8.26 With the aid of the interface schematic shown in Figure 8.22, explain the meaning/use of:
(i) the latency buffer,
(ii) 4B5B encoder/decoder,
(iii) clock synchronizer.

8.27 In relation to the timed token rotation protocol used with FDDI, explain the meaning of the terms:
(i) TTRT,
(ii) TRT,
(iii) THT,
(iv) early/late token.

efine the terms (i) maximum obtainable roughput and (ii) the maximum access delay a shared-medium network. Hence with the d of the example shown in Figure 8.25, stating clearly any assumptions you make, derive approximate formula for each in terms of e TTRT and ring latency, T_l.

8.7

erive the slot time in bits for a Fast Ethernet ub operating with drop cables of up to 100 m nd a bit rate of 100 Mbps. State the implications of your answer in relation to that of a 0BaseT hub.

ith the aid of the wiring configuration of a 00BaseT hub shown in Figure 8.26(a), xplain how the carrier sense and collision etection functions associated with CSMA/CD re carried out.

State why the 8B6T encoding scheme is used for transmissions over each twisted-pair wire with 100BaseT. Also explain why a DC balancing scheme is required.

With the aid of the diagrams shown in Figures 8.27 and 8.28, explain:

(i) how DC balance is maintained,
(ii) the latency associated with the 8B6T encoding sequence,
(iii) how additional error detection is obtained by sending one of two different end-of-stream codes,
(iv) how the start-of-stream code used gives a more reliable collision detection.

What is the difference between a repeater hub and a switching hub? With the aid of Figure 8.29, explain the operation of a Fast Ethernet switch. Include in your explanation how duplex working is achieved.

With the aid of the network schematic shown in Figure 8.30, explain how a Fast Ethernet switch can be used to provide faster access to site servers for stations attached to 10/100BaseT hubs.

8.35 In relation to a Gigabit Ethernet repeat explain how the shorter slot time is ove using

(i) carrier extension,
(ii) frame bursting.

8.36 Produce a diagram of a network configu similar to that shown in Figure 8.30 that s how a Gigabit Ethernet switch can be us interconnect several 100BastT hubs/swit to a server. Show clearly the bit rate used each interconnecting line.

Section 8.8

8.37 With the aid of the protocol framework sho in Figure 8.31, explain the role of the followi sublayer protocols:

(i) PMD,
(ii) convergence,
(iii) MAC, (iv) LLC.

8.38 With the aid of the interlayer paramet shown in Figure 8.35, explain the meaning use of:

(i) the DSAP and SSAP,
(ii) SNP including the ORG and type field
(iii) CTL.

Section 8.9

8.39 With the aid of the diagram of an enterp network shown in Figure 8.36, explain h intersite communications are carried out us the following types of intersite gateway:

(i) remote bridge,
(ii) IP/IPX router.

8.40 With the aid of the diagrams shown in Fig 8.37, state the role and describe the operat of an inverse multiplexer. Include in yo description the principles of the delay comp sation and resynchronization procedures.

8.41 With the aid of the network schematic sho in Figure 8.38, explain how a public fram relay network can be used to create an ente prise network. Hence use the frame form nd routing principles shown in Figure 8.39 to xplain how frames are relayed/routed etween all the network sites. Include in your xplanation the role of the following fields in ach frame header:

) DLCI,
i) FECN, BECN and DE bits.

xplain the use of a CCLM frame and the actions carried out if a frame is corrupted.

8.42 With the aid of the network schematic shown in Figure 8.40, explain how both telephony (via a PBX) and data (via a remote bridge or router) are extended to cover an entire enterprise. Include in your explanation the role and operation of:

(i) a subrate multiplexer,
(ii) a frame relay adapter.

Can the latter also be used for telephony?

9

e Internet

9.1 Introduction

As we saw in Chapter 1, the Internet is a global network that supports a va ety of interpersonal and interactive multimedia applications. A user ga access to these applications by means of an end system – normally referre as a host – which, typically, is a multimedia PC, a network computer, workstation. As we showed in Figure 1.2, the Internet comprises a la number of different access networks which are interconnected togethe means of a global internetwork. Associated with each access network network, intranet, enterprise network, site/campus LAN, and so on **gateway** and the global internetwork consists of an interconnected se regional, national, and international, networks all of which are interc nected together using high bit rate leased lines and devices known as **rou** **gateways** or simply **routers**.

The Internet operates in a packet-switched mode and Figure 9.1 sho the protocol stack associated with it. In the figure, we assume the netwo interface card in all hosts that are attached to an access network commu cate with other hosts using the TCP/IP protocol stack. As we showed Figure 5.11 (and explained in the accompanying text) this is not always th

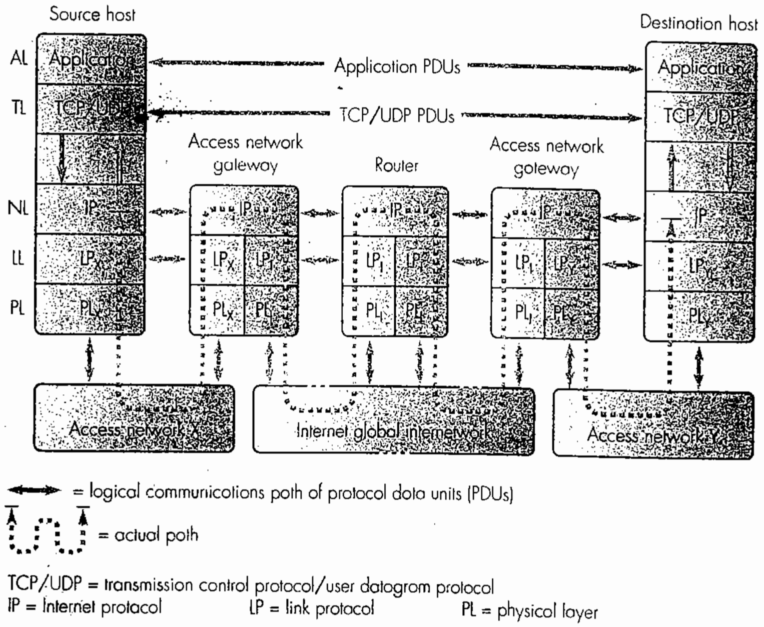

Source host
Destination host
AL
TL
NL
LL
PL
Application
TCP/UDP
IP
LPx
PLx
Application PDUs
TCP/UDP PDUs
Access network gateway
Router
Access network gateway
IP
LPx
LP1
PLx
PL1
LP1
LP2
PL1
PL2
LPY
PLY
Access network X
Internet global internetwork
Access network Y
= logical communications path of protocol data units (PDUs)
= actual path
TCP/UDP = transmission control protocol/user datagram protocol
IP = Internet protocol
LP = link protocol
PL = physical layer

Figure 9.1 Internet networking components and protocols.

case. Nevertheless, any end system (host) that communicates directly over the Internet – the email server in Figure 5.11 for example – does so using the TCP/IP protocol stack.

In general, the various access networks have different operational parameters associated with them in terms of their bit rate, frame format, maximum frame size, and type of addresses that are used. For example, in the case of a site/campus LAN, as we saw in the last chapter, a token ring LAN uses a different bit rate, frame format, and maximum frame size from an Ethernet LAN. This means, therefore, that since bridges can only be used to interconnect LAN segments of the same type, they cannot be used to perform the network interconnection function. Hence instead, the routing and forwarding operations associated with a gateway are performed at the network layer. In the TCP/IP protocol stack the network layer protocol is the **Internet protocol** (**IP**) and, as we show in Figure 9.1, in order to transfer packets of information from one host to another, it is the IP in the two hosts, together with the IP in each Internet gateway and router involved, that perform the routing and other harmonization functions necessary.

The IP in each host (that communicates directly over the Internet) has a unique Internet-wide address assigned to it. This is known as the host's

Internet address or, more usually, its **IP address**. Each IP address has
parts: a **network identifier (netid)** and a **host identifier (hostid)**. The all
tion of netids is centrally managed by the **Internet Network Informa**
Center (InterNIC) and each access network has a unique netid **assigned**
For example, each campus/site LAN is assigned a single netid. Th
address of a host attached to an access network then contains the uni
netid of the access network and a unique hostid. As with netids, hostids
centrally allocated but this time by the local administrator of the access
work to which the host is attached.

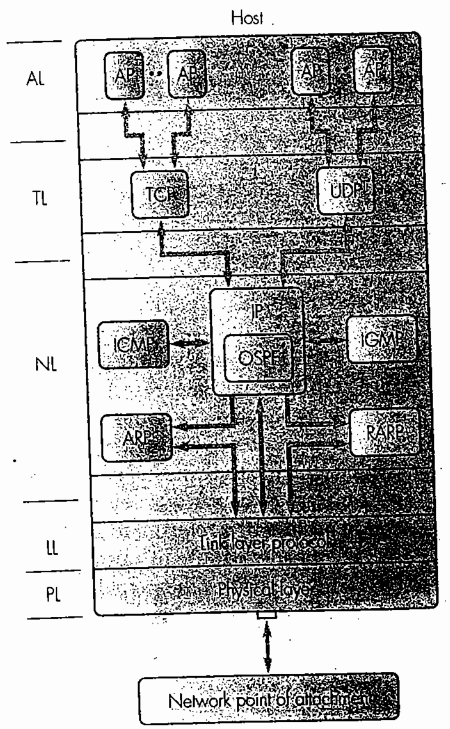
The IP provides a connectionless best-effort service to the transport la
above it which, as we show in the figure, is either the transmission con
protocol (TCP) or the user datagram protocol (UDP). Hence when eit
protocol has a block of information to transfer, it simply passes the block
its local IP together with the IP address of the intended recipient. T
(source) IP first adds the destination and source IP addresses to the head
the block, together with an indication of the source protocol (TCP or UD
to form what is known as an **IP datagram**. The IP then forwards the datagra
to its local gateway. At this point the datagram is often referred to as a **pack**
and hence the two terms are used interchangeably.

Each access gateway is attached to an internetwork router and, at regu
intervals, the IP in these routers exchange routing information. When this
complete, each router has built up a **routing table** which enables it to rout
packet/datagram to any of the other networks/netids that make up
Internet. Hence, on receipt of a packet, the router simply reads the desti
tion netid from the packet header and uses the contents of its routing ta
to forward the packet on the path/route through the global internetwo
first to the destination internetwork router and, from there, to the dest
tion access gateway. Assuming the size of the packet is equal to or less t
the maximum frame size of the destination access network, on receipt of
packet, the destination gateway reads the hostid part of the destinatio
address and forwards the packet to the local host identified by the ho
part. The IP in the host then strips off the header from the packet and pa
the block of information contained within it – known as the **payload** – to
peer transport layer protocol indicated in the packet header.

If the size of the packet is greater than the maximum frame size – th
the **maximum transmission unit** (MTU) – of the destination access netw
the IP in the destination gateway proceeds to divide the block of informa
contained in the packet into a number of smaller blocks each known
fragment. Each fragment is then forwarded to the IP in the destination h
in a separate packet the length of which is determined by the MTU of
access network. The destination IP then reassembles the fragments of in
mation from each received packet to form the original submitted block
information and passes this to the peer transport layer protocol indicated
the packet header.

As we shall see in the following sections, the above is just a summary of the operation of the IP and, in practice, in order to perform the various functions we have just described, the IP uses a number of what are known as **adjunct protocols.** These are identified in Figure 9.2 and a summary of the role of each protocol is as follows:

- **The address resolution protocol (ARP)** and the **reverse ARP (RARP)** are used by the IP in hosts that are attached to a broadcast LAN (such as an Ethernet or token ring) in order to determine the physical (MAC) address of a host or gateway given its IP address (ARP) and, in the case of the RARP, the reverse function.



AP = application protocol/process
IP = Internet protocol
ARP = address resolution protocol
RARP = reverse ARP
ICMP = Internet control message protocol
IGMP = Internet group message protocol
OSPF = open shortest path first

Figure 9.2 IP adjunct protocols.

- **The open shortest path first (OSPF)** protocol is an example of a routing protocol used in the global internetwork. Such protocols are present in each internetwork router and are utilized to build up the contents of the routing table that is used to route packets across the global internetwork
- The **Internet control message protocol (ICMP)** is used by the IP in a hos or gateway to exchange error and other control messages with the IP in another host or gateway.
- The **Internet group management protocol (IGMP)** is used with multicasting to enable a host to send a copy of a datagram to the other hosts that are part of the same multicast group.

In this chapter we explain the operation of the different parts of the IP an each of the adjunct protocols in some detail. Also, we describe the structure o the Internet global internetwork in more detail as this influences the overal routing strategy and routing protocols that are used. Currently, the most widel used version of the IP is version 4 and hence most of the chapter is devoted t this. In a longer time span, however, this will be replaced by version 6. Henc we describe the main features of this too and how it differs and interoperate with version 4. We shall defer discussion of the two transport layer protocol until Chapter 12 when we describe application protocols and related issues.

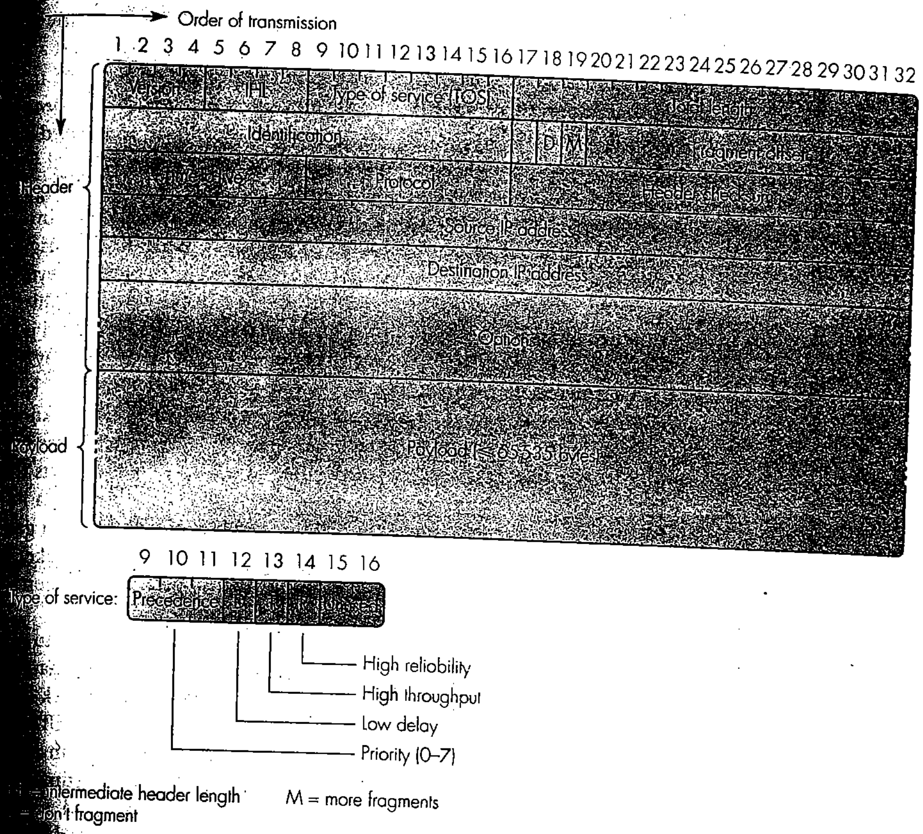
9.2 IP datagrams

As we indicated in the introduction, the IP is a connectionless protocol a all user information is transferred in the payload part of what is known a datagram. The header of each datagram contains a number of fields format of which are shown in Figure 9.3.

The *version* field contains the version of the IP used to create the d gram and ensures that all systems – gateways, routers, and hosts – that pro the datagram/packet during its transfer across the Internet to the destina host interpret the various fields correctly. The current version number and hence the IP is referred to as **IP version 4** or simply **IPv4**.

The header can be of variable length and the *intermediate header l* (*IHL*) field specifies the actual length of the header in multiples of 3 words. The minimum length (without options) is 5. If the datagram con options, these are in multiples of 32 bits with any unused bytes filled **padding** bytes. Also, since the IHL field is 4 bits, the maximum permis length is 15.

The *type of service (TOS)* field allows an application protocol/proces specify the relative priority (precedence) of the application data and the ferred attributes associated with the path to be followed. It is used by gateway and router during the transmission and routing of the packe transmit packets of higher priority first and to select a line/route that has specified attributes should a choice be available. For example, if a route w


Order of transmission
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Version
IHL
Type of service (TOS)
Total length
Identification
D
M
Fragment offset
Protocol
Header checksum
Source IP address
Destination IP address
Options
Payload (≤ 65 535 bytes)
Header
Payload
9 10 11 12 13 14 15 16
Type of service:
Precedence
High reliability
High throughput
Low delay
Priority (0–7)
intermediate header length
don't fragment
M = more fragments

3 **IP datagram/packet format and header fields.**

a minimum delay is specified then, given a choice of routes, the line with the smallest delay associated with it should be chosen. We shall discuss the use of the TOS field further in Section 9.8.

The *total length* field defines the total length of the initial datagram including the header and payload parts. This is a 16-bit field and hence the maximum length is 65 535 (64K – 1) bytes and, as we explain in the next section, should the contents of the initial datagram need to be transferred in multiple (smaller) packets, then the value in *total length* is used by the destination host to reassemble the payload contained within each smaller packet – known as a fragment – into the original payload. In addition, each smaller packet contains the same value in the *identification* field to enable the destination host to relate each received packet fragment to the same original datagram.

The next three bits are known as *flag bits* of which two are currentl
The first is known as the *don't fragment* or *D-bit*. It is set by a source hos
examined by routers. A set D-bit indicates that the packet should be
ferred in its entirety or not at all. The second is known as *more fragm*
M-bit and this also is used during the reassembly procedure associate
data transfers involving multiple smaller packets/fragments. It is set to
but the last packet/fragment in which it is set to 0. In addition, the *fra*
offset is used by the same procedure to indicate the position of the first b
the fragment contained within a smaller packet in relation to the ori
packet payload. All fragments except the last one are in multiples of 8 by

The value in the *time-to-live* field defines the maximum time for wh
packet can be in transit across the Internet. The value is in seconds and
by the IP in the source host. It is then decremented by each gateway
router by a defined amount and, should the value become zero, the pack
discarded. In principle, this procedure allows a destination IP to wait a kn
maximum time for an outstanding packet fragment during the reassem
procedure. In practice, it is used primarily by routers to detect packets
are caught in loops. For this reason, therefore, the value is normally a
count. In this case, the *hop count* value is decremented by one by each ga
way/router visited and, should the value become zero, the packe
discarded. We shall identify how looping can occur in Section 9.6 when
discuss the subject of routing.

The value in the *protocol* field is used to enable the destination IP to
the payload within each received packet to the same (peer) protocol that
the data. As we showed in Figure 9.2, this can be an internal network la
protocol such as the ICMP, or a higher-layer protocol such as TCP or UD

The *header checksum* applies just to the header part of the datagram an
a safeguard against corrupted packets being routed to incorrect destinatio
It is computed by treating each block of 16 bits as an integer and add
them all together using 1s complement arithmetic. As we showed in
example in Figure 6.20(b), the checksum is then the complement (inve
of the 1s complement sum.

The *source address* and *destination address* are the Internetwide IP addre
of the source and destination host respectively.

Finally, the *options* field is used in selected datagrams to carry addit
information relating to:

- *security:* the payload may be encrypted for example, or be made accessible only to a specified user group. The *security* field then conta
fields to enable the destination to decrypt the payload and authentica
the sender;
- *source routing:* if known, the actual path/route to be followed through
Internet may be specified in this field as a list of gateway/router addresses;
- *loose source routing:* this can be used to specify preferred routers in a pat

---

- *route recording:* this field is used by each gateway/router visited during the passage of a packet through the Internet to record its address. The resulting list of addresses can then be used, for example, in the source routing field of subsequent packets;
- *stream identification:* this, together with the source and destination addresses in the datagram header, enables each gateway/router along the path followed by the packet to identify the stream/flow to which the packet belongs and, if necessary, give the packet precedence over other packets. Examples include streams containing samples of speech or compressed video;
- *time-stamp:* if present, this is used by each gateway/router along the path followed by the packet to record the time it processed the packet.

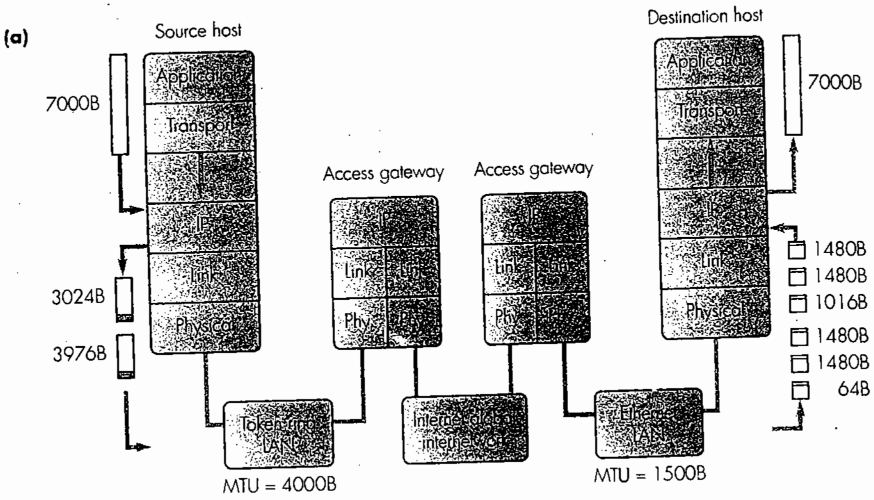
9.3 Fragmentation and reassembly

As we explained in Section 9.1, if the size of a packet is greater than the MTU of the destination access network – or an intermediate network in the global internetwork – the IP in the destination gateway – or intermediate router – divides the information received in the packet into a number of smaller blocks known as fragments. Each fragment is then forwarded to the IP in the destination host in a separate packet the length of which is determined by the MTU of the access/intermediate network. The IP in the destination host then reassembles the fragments of information from each received packet to form the original submitted block of information. It then passes this to the peer transport layer protocol indicated in the protocol field of the packet header.

To see how the various fields in each packet header are used to perform this function, consider the transport protocol in a host that is attached to a token ring LAN transferring a block of 7000 bytes – including the transport protocol header – over the Internet to the transport protocol in a host that is attached to an Ethernet LAN. Let us assume that the MTU associated with the token ring LAN is 4000 bytes and that of the Ethernet LAN 1500 bytes. Also that the header of each IP datagram requires 20 bytes. The steps taken to transfer the complete block of 7000 bytes are shown in Figure 9.4(a).

Since the header of each datagram requires 20 bytes, the maximum usable data in each token ring frame is 4000 – 20 = 3980 bytes. Similarly, that in each Ethernet frame is 1500 – 20 = 1480 bytes. However, all fragments of user data (except the last one) must be in multiples of 8 bytes. So we shall have to limit the maximum user data in each packet transferred over the token ring to 3976 bytes. In the case of the Ethernet, 1480 is divisible by 8 and so this value can be used unchanged.

To transfer the block of 7000 bytes over the token ring LAN requires two datagrams, one with 3976 bytes of user data and the second 7000 – 3976 = 3024 bytes. The values for the various fields associated with the fragmentation and reassembly procedures in each datagram header are given in Figure 9.4(b).



Note: All values shown are the amounts of user data in each packet/frame in bytes

(b) *Token ring LAN:*

	(i)	*(ii)*
Identification	20	20
Total length	7000	7000
Fragment offset	0	497
(User data)	3976	3024
M-bit	1	0

(c) *Ethernet LAN:*

	(i)	*(ii)*	*(iii)*	*(iv)*	*(v)*	*(vi)*
Identification	20	20	20	20	20	20
Total length	7000	7000	7000	7000	7000	7000
Fragment offset	0	185	370	497	682	867
(User data)	1480	1480	1016	1480	1480	64
M-bit	1	1	1	1	1	0

Figure 9.4 Fragmentation and reassembly example: (a) Internet schematic; (b) packet header fields for token ring LAN; (c) Ethernet LAN.

The value in the *identification* field is the same in all fragments and is us by the destination IP to relate each fragment to the same original bloc information. In the example, we assume a value of 20 has been allocated the IP in the source host.

The *total length* is the number of bytes in the initial datagram includ the 20-byte header. However, since we have subtracted 20 from the maxim user data value associated with each LAN, we have shown this as 7000. No that this is the same in all the datagram fragments and hence the destinati IP can readily determine when all fragments have been received. Th

fragment offset then indicates the position of the user data in each fragment – in multiples of 8 bytes – relative to the start of the initial datagram. Finally, the *more fragments* (*M*) *bit* is 1 in each fragment and 0 in the final fragment.

We assume that the two datagrams/packets created by the source IP are transferred over the global internetwork unchanged and, on reaching the access gateway attached to the Ethernet LAN, the smaller maximum user data value of 1480 bytes means that both packets must be further fragmented. As we show in Figure 9.4(c), both packets must be fragmented into three smaller packets. The first into two maximum sized packets of 1480 bytes and a further packet containing 3976 – 2(1480) = 1016 bytes, and the second containing two maximum sized packets and a further packet containing 3024 – 2(1480) = 64 bytes. The IP in the destination host then reassembles the user data in each of the six packets it receives into the original 7000-byte block of information and delivers this to the peer transport protocol.

As we explained, the *time-to-live* field in each packet header – and fragment header – is present to avoid packets endlessly looping around the Internet (normally as a result of routing table inconsistencies) and also to set a maximum limit on the time a host needs to wait for a delayed/corrupted/discarded datagram fragment. Hence although the use of fragmentation would appear to be relatively straightforward, there are drawbacks associated with its use. For example, as we shall explain in Section 12.3.2, with the TCP transport protocol, if an acknowledgment of correct receipt of a submitted block is not received within a defined maximum time limit, the source TCP will retransmit the complete block. Thus, as we can deduce from the example in Figure 9.4, it only needs one of the six datagram fragments to be delayed or discarded to trigger the retransmission of the complete 7000-byte block. As a result, therefore, most TCP implementations avoid the possibility of fragmentation occurring by limiting the maximum submitted block size – including transport protocol header – to 1048 bytes or, in some instances, 556 bytes. Alternatively, as we shall expand upon in Section 9.7, it is possible for the source IP, prior to sending any transport protocol (user) data, to determine the MTU of the path to be followed through the Internet. Then, if this is smaller than the submitted user data, the source IP fragments the data using this MTU. In this way, no further fragmentation should be necessary during the transfer of the packets through the global internetwork.

9.4 IP addresses

Each host, gateway, and router has a unique Internetwide IP address assigned to it that comprises a netid and a hostid part. Hence in the case of gateways and routers that interconnect two (or more) networks together, each gateway/router port – also referred to as an interface – has a different netid associated with it. In order to give the Internet Network Information Center some flexibility when assigning netids, one of three different address formats

can be used. Each format is known by an **address class** and, as we sho
Figure 9.5, classes A, B, and C are all different types of unicast addresses.
Each of these classes is intended for use with a different size of netw
For example, at one extreme a large national network and at the othe
small site LAN. The class to which an address belongs can be determin
from the position of the first zero bit in the first four bits. The remaining b
then specify the netid and hostid parts with the boundary separating the t
parts located on byte boundaries to simplify decoding.

Class A addresses have 7 bits for the netid and 24 bits for the hostid; cl
B addresses have 14 bits for the netid and 16 bits for the hostid; and clas
addresses have 21 bits for the netid and 8 bits for the hostid. Class A addres
are intended for use with networks that have a large number of attach
hosts (up to 2^{24}) while class C addresses allow for a large number of netwo
each with a small number of attached hosts (up to 256). An example of a cl
A network is a large national network and an example of a class C network
a small site LAN.

Netids and hostids comprising either all 0s or all 1s have special meani

- An address with a hostid of all 0s is used to refer to the network in the
netid part rather than a host.
- An address with a netid of all 0s implies the same network as the source
network/netid.

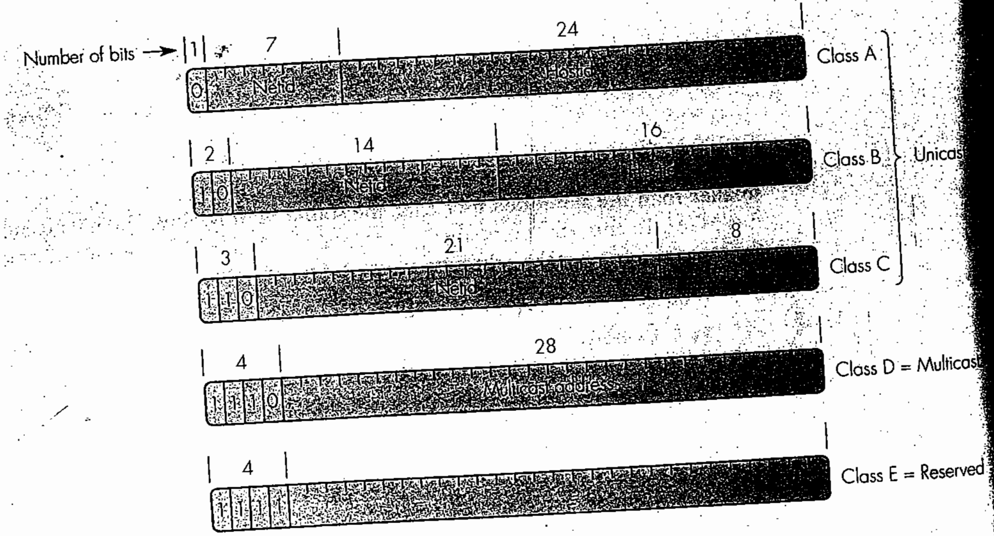


Figure 9.5 IP address formats.

- An address of all 1s means broadcast the packet over the source network.
- An address with a hostid of all 1s means broadcast the packet over the destination network in the netid part.
- A class A address with a netid of all 1s is used for test purposes within the protocol stack of the source host. It is known, therefore, as the **loopback address.**

To make it easier to communicate IP addresses, the 32 bits are first broken into four bytes. Each byte is then converted into its equivalent decimal form and the total IP address is represented as the four decimal numbers with a dot (period) between each. This is known as the **dotted decimal notation** and some examples are as follows:

00001010 00000000 00000000 00000000 = 10.0.0.0.
= class A, netid 10

10000000 00000011 00000010 00000011 = 128.3.2.3
= class B, netid 128.3, hostid 2.3

11000000 00000000 00000001 11111111 = 192.0.1.255
= class C, all hosts broadcast on netid 192.0.1

Example 9.1

Assuming the IP address formats shown in Figure 9.5, derive the range of host addresses for classes A, B, and C. Give your answer in dotted decimal notation and also straight decimal.

Answer:

Class A:
Netid = 1 to 127
= 126 networks
Hostid = 0.0.0 to 255.255.255
= 16 777 214 hosts

Class B:
Netid = 128.0 to 191.255
= 16 382 networks
Hostid = 0.0 to 255.255
= 65 534 hosts

Class C:
Netid = 192.0.0 to 223.255.255
= 2 097 152 networks
Hostid = 0 to 255
= 254 hosts

Note that we have not used hostids of all 0s or all 1s.

Class D addresses are reserved for multicasting. As we explained in Section 8.2, in a LAN a frame can be sent to an individual, broadcast, or group address. The group address is used by a station to send a copy of the frame to all stations that are members of the same multicast group (and hence have the same multicast address). In the case of LANs, the group address is a MAC address and the class D IP address format is provided to enable this mode of working to be extended over the complete Internet.

As we can see, unlike the three unicast address classes, the 28-bit multicast group address has no further structure. As we shall expand upon in Section 9.6.9, multicast group addresses are assigned by the **Internet assigned numbers authority** (*IANA*). Although most of these are assigned dynamically (for conferences and so on), some are reserved to identify specific groups of hosts and/or routers. These are known as **permanent multicast group addresses.** Examples are 224.0.0.1 which means all hosts (and routers) on the same broadcast network, and 224.0.0.2 which means all routers on the same site network.

9.4.1 Subnets

Although the basic structure shown in Figure 9.5 is adequate for most addressing purposes, the introduction of multiple LANs at each site can mean unacceptably high overheads in terms of routing. As we described in Section 8.5, MAC bridges are used to interconnect LAN segments of the same type. This solution is attractive for routing purposes since the combined LAN then behaves like a single network. When interconnecting dissimilar LAN types, as we explained in Section 8.8.4, the differences in frame format and, more importantly, frame length, mean that routers are normally used since the fragmentation and reassembly of packets/frames is a function of the network layer rather than the MAC sublayer. However, the use of routers means that, with the basic address formats, each LAN must have its own netid. In the case of large sites, there may be a significant number of such LANs.

This means that with the basic addressing scheme, all the routers relating to a site need to take part in the overall Internet routing function. As we shall expand upon in Section 9.5, the efficiency of any routing scheme is strongly influenced by the number of routing nodes that make up the Internet. The concept of **subnets** has been introduced to decouple the routers – and hence routing – associated with a single site from the overall routing function in the global internetwork. Essentially, instead of each LAN associated with a site having its own netid, only the site is allocated a netid. Each LAN is then known as a **subnet** and the identity of each (LAN) subnet then forms part of the hostid field. This refined address format is shown in Figure 9.6(a).

The same address classes and associated structure are used, but the netid now relates to a complete site rather than to a single subnet. Hence, since only a single gateway/router attached to a site performs internetwide routing, the netid is considered as the **Internet part.** For a single netid with a number of associated subnets, the hostid part consists of two subfields: a **subnetid part** and a **local hostid part**. Because these have only local significance, they are known collectively as the **local part.** Also, to discriminate between the routers in the global internetwork and those in a local site network, the latter are known as **subnet routers.**

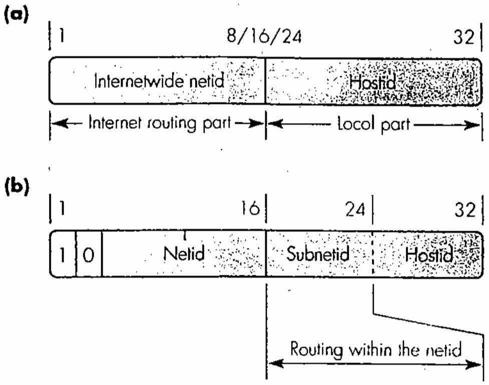


Figure 9.6 Subnet addressing: (a) address structure; (b) example.

Because of the possibly wide range of subnets associated with different site networks, no attempt has been made to define rigid subaddress boundaries for the local address part. Instead, an **address mask** is used to define the subaddress boundary for a particular network (and hence netid). The address mask is kept by the site gateway and all the subnet routers at the site. It consists of binary 1s in those bit positions that contain a network address – including the netid and subnetid – and binary 0s in positions that contain the hostid. Hence an address mask of

11111111 11111111 11111111 00000000

means that the first three bytes (octets) contain a network/subnet identifier and the fourth byte contains the host identifier.

For example, if the mask relates to a class B address – a zero bit in the second bit position – this is readily interpreted as: the first two bytes are the internetwide netid, the next byte the subnetid, and the last byte the hostid on the subnet. Such an address is shown in Figure 9.6(b).

Normally, dotted decimal is used to define address masks and hence the above mask is written:

255.255.255.0

Byte boundaries are normally chosen to simplify address decoding. So wi
this mask, and assuming the netid was, say, 128.10, then all the hosts attach
to this network would have this same netid. In this way, the presence of a p
sibly large number of subnets and associated (subnet) routers is transpare
to all the other Internet gateways and routers for routing purposes.

Example 9.2

The administrator of a campus LAN is assigned a single class B
address of 150.10.0.0. Assuming that the LAN comprises 100 subne
each of which is connected to an FDDI backbone network using
subnet router, define a suitable address mask for the site if the max
mum number of hosts connected to each subnet is 70.

Answer:

A class B IP address means that both the netid part and the local pa
are each 16 bits. Hence the simplest way of meeting this requirement
to divide the local part into two; 8 bits for the subnetid and 8 bits for th
hostid.

This will allow for up to 254 subnets and 254 hosts per subnet, igno
ing all 1s and all 0s.

The address mask, therefore, is 255.255.255.

9.5 ARP and RARP

As we outlined in Section 9.1, the address resolution protocol (ARP) and th
reverse ARP (RARP) are used by the IP in hosts that are attached to a broa
cast LAN. The ARP is used to determine the MAC address of another host
gateway that is attached to the same LAN given the IP address of the h
gateway. It is defined in **RFC 826**. The RARP performs the reverse functi
and is defined in **RFC 903**. We explain the operation of each separately.

9.5.1 ARP

Associated with each host are two addresses: its IP address and its MA
address which, since it is assigned to the MAC integrated circuit when it
manufactured, is known also as the host's hardware (or physical) addre
Normally, both addresses are stored in the configuration file of the host o
the hard disk. In order to describe the operation of the ARP, we shall use th
LAN topology shown in Figure 9.7.

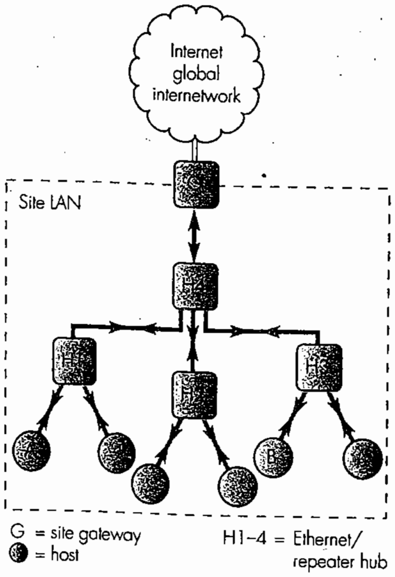


Figure 9.7 Example topology for describing the operation of the ARP.

As we can see, this comprises three Ethernet hubs (H1, H2, and H3) that are interconnected by means of a fourth hub (H4). There is also a connection between H4 and the site gateway (G). We assume that all the hubs are simple repeater hubs and that all hosts have just been switched on and hence have sent no frames. Associated with each ARP is a routing table known as the **ARP cache.** This contains a list of the IP/MAC address-pairs of those hosts with which host A has recently communicated and, when the host is first switched on, it contains no entries. First we shall explain the steps taken by the ARP in host A to send a datagram to another host on the same LAN – host B – and then to a host on a different LAN via the gateway.

On receipt of the first datagram from the IP in host A, the ARP in A reads the destination IP address of B contained in the datagram header and determines it is not in its cache. Hence it broadcasts an **ARP request message** in a broadcast frame over the LAN and waits for a reply. The request message contains both its own IP/MAC address-pair and the (target) IP address of the destination, host B. Being a broadcast frame this is received by the ARP in all hosts attached to the LAN.

The ARP in host B recognizes its own IP address in the request message and proceeds to process it. It first checks to see whether the address-pair of the source is within its own cache and, if not, enters them. This is done since it is highly probable that the destination host will require the MAC address of the source when the higher-layer protocol responds to the message contained within the datagram payload. The ARP in host B then responds by returning an **ARP reply** message (containing its own MAC address) to the ARP in host A using the latter's MAC address contained in the request message. On receipt of the reply message, the ARP in host A first makes an entry of the requested IP/MAC address-pair in its own cache and then passes the waiting datagram to either the LLC sublayer (if one is present) or (if not) to the MAC sublayer together with the MAC address of host B which indicates where it should be sent. At B the datagram is passed directly to the IP for processing.

Being on the same broadcast network as all the site hosts, the LAN port of the gateway receives a copy of all broadcast frames containing ARP request and reply messages. On receipt of each message, the ARP first checks to see if it has the IP/MAC address-pair(s) contained in the message in its cache and if not, adds them to the cache. In this way, the site gateway learns the address-pair of all the hosts that are attached to the site LAN.

To send a datagram from, say, host A to a host on a different LAN – and hence netid – the ARP in A broadcasts the request message as before. In this case, however, on receipt of the message, the gateway determines that the netid part of the destination IP address relates to a different network and responds by returning a reply message containing its own address-pair. Hence A makes an entry of this in its cache and proceeds to forward the datagram to the gateway as if it was the destination host. The gateway then forwards the datagram/packet over the Internet using one of the global internetwork routing protocols we shall describe later in Section 9.6. The ARP in the gateway is known as a **proxy ARP** since it is acting as an agent for the ARP in the destination host.

When the gateway receives the response packet from the destination host, it reads the destination IP address from the header – host A – and obtains from its cache the MAC address of A. It then transfers the packet to the IP in A using the services provided by the MAC sublayer. A similar procedure is followed for bridged LANs and for router-based LANs except that with the latter, the ARP in the subnet router that is connected to the same subnet as the source host acts as the proxy ARP. Also, in order to allow for hosts to change their network point of attachments, entries in the ARP cache timeout after a predefined time interval.

9.5.2 RARP

As we indicated at the start of the last section, normally the IP/MAC address-pair of a host is stored in the configuration file of the host on its hard disk. With diskless hosts, clearly this is not possible and hence the only address that is known is the MAC address of the MAC chipset. In such cases, therefore, the alternative reverse address resolution protocol (RARP) is used.

The server associated with a set of diskless hosts has a copy of the IP/MAC address-pair of all the hosts it serves in a configuration file. When a diskless host first comes into service, it broadcasts a **RARP request** message containing the MAC address of the host onto its local LAN segment. Being a broadcast, the server receives this and, on determining it is a RARP message, the MAC/LLC sublayer passes the message to the RARP. The latter first uses the MAC address within it to obtain the related IP address from the configuration file and then proceeds to create a **RARP reply** message containing the IP address of the host and also its own address-pair. The server then sends the reply message back to the host and, once its own address-pair is known, the ARP in the diskless host can proceed as before.

9.5.3 ARP/RARP message formats and transmission

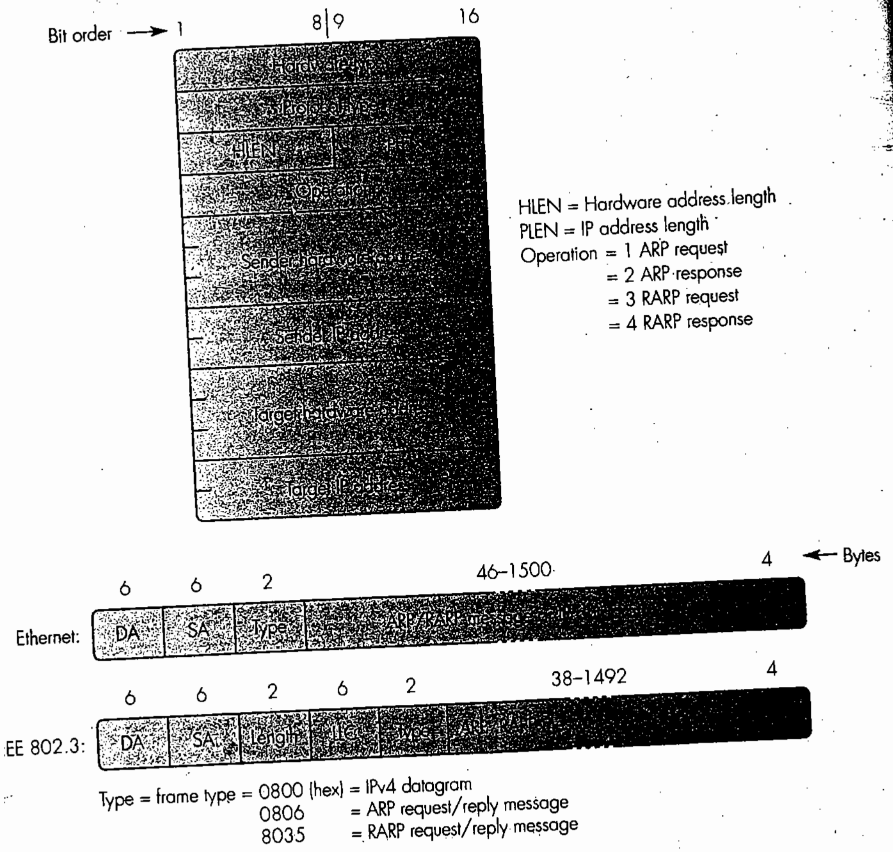
As we show in Figure 9.8(a), the format of the ARP and RARP request and reply messages are the same both having a fixed length of 28 bytes. Note that the term "hardware address" is used to refer to a MAC address and "target" the recipient of a request.

The *hardware type* field specifies the type of hardware (MAC) address contained within the message, for example 0001 (hex) in the case of an Ethernet. The ARP and RARP can be used with other network protocols (as well as the IP) and the *protocol type* indicates the type of network address being resolved; for example, 0800 (hex) is used for IP addresses. The *HLEN* and *PLEN* fields specify the size in bytes of the hardware (MAC) and protocol (IP) address lengths respectively, for example 06 (hex) for an Ethernet and 04 (hex) for the IP. The *operation* field indicates whether the message (resolution operation) is an ARP request (0001) or reply (0002) or a RARP request (0003) or reply (0004).

The next four fields specify the hardware (MAC)/IP address-pair of the sender (source) and the target (destination). For example, in an ARP request message just the address-pair of the sender is used while in the reply message all four addresses are used.

As we indicated in the introduction, both the ARP and RARP are integral parts of IP inasmuch as, once the MAC address relating to an IP address is present in the ARP cache, the IP can use this to initiate the transmission of a datagram to its intended recipient directly. Hence, as we can deduce from Figure 9.2, on receipt of a frame, the receiving MAC/LLC sublayer must be able to determine to which protocol the frame contents should be sent: IP, ARP, or RARP. To achieve this, when each of the protocols passes a message/datagram to the LLC/MAC sublayer for transmission, in addition to the MAC address of the intended recipient, it specifies the name of the (peer) protocol in the destination (IP/ARP/RARP) to which the message/datagram should be passed.

As we show in Figure 9.8(b), this is specified in a two-byte *type* field which immediately precedes the message/datagram in the user data field of the MAC frame. As we explained in Section 8.8.3, normally, the LLC sublayer is not used with the original Ethernet MAC standard and hence the *type* field


Bit order
1
8|9
16
Hardware type
Protocol type
HLEN
PLEN
Operation
Sender hardware address
Sender IP address
Target hardware address
Target IP address
HLEN = Hardware address length
PLEN = IP address length
Operation = 1 ARP request
= 2 ARP response
= 3 RARP request
= 4 RARP response
Bytes
6
6
2
46–1500
4
Ethernet:
DA
SA
Type
ARP/RARP message
6
6
2
6
2
38–1492
4
IEEE 802.3:
DA
SA
Length
LLC
Type
ARP/RARP
Type = frame type = 0800 (hex) = IPv4 datagram
0806 = ARP request/reply message
8035 = RARP request/reply message

9.8 ARP and RARP message formats and transmission: (a) ARP and RARP message ts; (b) MAC frame format with Ethernet and IEEE802.3.

immediately follows the MAC *source address* (*SA*). With the more recent IEE 802.3 MAC standard, however, the LLC sublayer is present and hence the *typ* field immediately follows the 6 bytes required for the LLC protocol. However the maximum frame length associated with the 802.3 standard (1500 bytes) means that the length indicator value in the header is always different from the three *type* field values. Hence the receiving MAC sublayer can readily determine which of the two standards is being used and, therefore, where the *type* field is located. With a token ring LAN, since there is only one standard this problem does not arise.

xample 9.3

Determine the amount of padding required in a [illegible] transmitting an ARP/RARP message over (i) an Ethe[illegible] an IEEE 802.3 LAN.

Answer:

As we explained in Section 9.5.3, each ARP/RARP message [illegible] 28 bytes. Also, as we explained in Section 8.3, the minimum frame [illegible] associated with the CSMA/CD MAC method is 512 bits (64 bytes) including the FCS.

(i) With an Ethernet LAN, the header plus FCS requires 18 bytes and hence the minimum size of the user data field is (64 − 18) = 46 bytes. Thus the number of pad bytes required = (46 − 28) = 18 bytes.

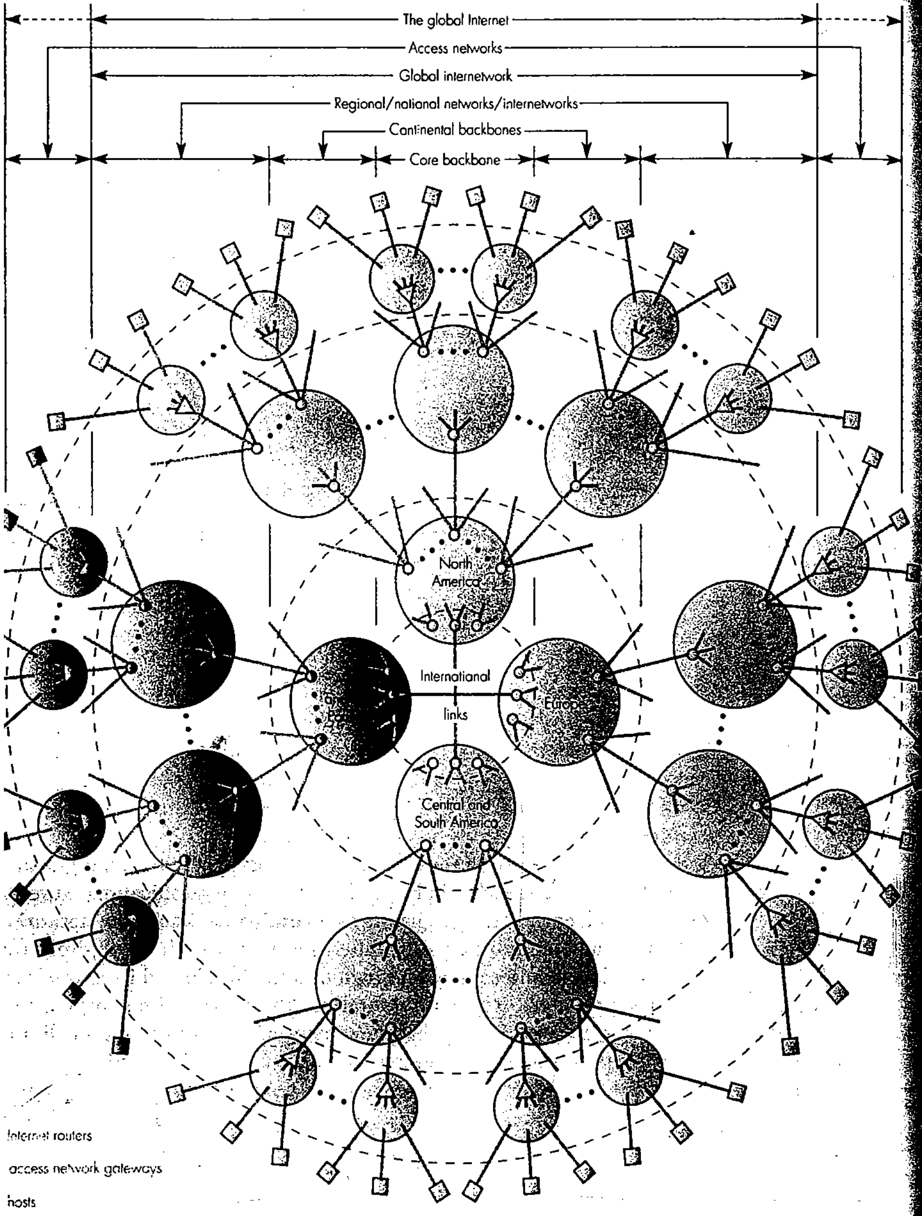
(ii) With an IEEE 802.3 LAN, the header plus FCS requires 26 bytes and hence the minimum size of the user data field is (64 − 26) = 38 bytes. Thus the number of pad bytes required = (38 − 28) = 10 bytes.

9.6 Routing algorithms

The Internet comprises many thousands of access networks that are geographically distributed around the world. As a result, the global internetwork used to interconnect the gateways associated with these (access) networks is not a single network as such but rather a collection of different types of network that have evolved over the lifetime of the Internet. The general architecture of the Internet is shown in Figure 9.9.

As we can see, the various networks making up the Internet global internetwork we showed earlier in Figure 9.1 are in a hierarchical structure. At the highest level, each continent has its own **continental backbone network**. This comprises a geographically distributed set of very high throughput routers which are interconnected by very high bit rate lines leased from one or more network providers. The continental backbones are then interconnected together by means of a **core backbone network** consisting of a small number of very high throughput (intercontinental) routers and leased lines.

At the second level, attached to all of the other routers in each continental backbone are a number of regional and national networks and, in some instances, large internetworks. In general, these also consist of a geographically distributed set of high throughput routers interconnected by high bit rate leased lines. Because of the historical evolution of the Internet, however, in addition, there are a number of legacy networks that operate using a different protocol stack from TCP/IP.


The global Internet
Access networks
Global internetwork
Regional/national networks/internetworks
Continental backbones
Core backbone
North America
International links
Europe
Central and South America
Internet routers
access network gateways
hosts

re 9.9 General architecture of the global Internet.

The various types of access networks – site/campus LANs, ISP networks, enterprise networks – are then attached to one of the regional or national networks/internetworks by means of either an **IP access network gateway** or, if the access network uses a different protocol stack from TCP/IP, a **multiprotocol gateway.**

All routing within the total global internetwork is carried out by the IP in each router using the netid part of the destination IP address in each datagram header. In practice, as we shall see in the following subsections, there are only a small number of different routing algorithms used and, in order to explain their principle of operation, we shall use the simple internetwork topology shown in Figure 9.10.

As we can see, each of the routers in the interconnection network has a number of access networks attached to it by means of (access) gateways. We assume that each access network is a site/campus LAN with a single netid and that the ARP in each host is used to carry out routing within the access network. Hence, as we explained in Section 9.5.1, the ARP in each gateway acts as an agent (proxy ARP) on behalf of all hosts at the site/campus to relay packets to and from the interconnection network.

The interconnection network itself comprises four routers (R1 – R4) that are interconnected by, say, leased lines. For description purposes, each line has a pair of numbers associated with it. The first we shall use as a line identifier and the second is what is referred to as the **cost** of the line. For example, the cost could be based on the line bit rate and, normally, the higher the line

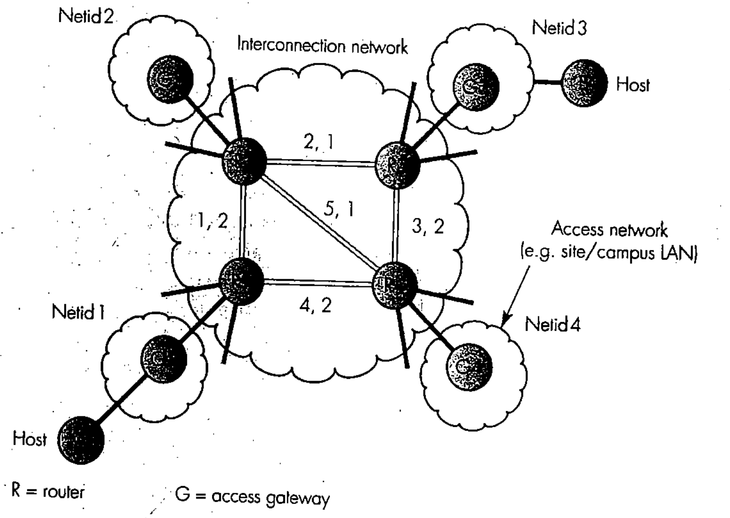

Netid 2
Interconnection network
Netid 3
Host
2, 1
5, 1
1, 2
3, 2
Access network
(e.g. site/campus LAN)
4, 2
Netid 1
Netid 4
Host
R = router
G = access gateway

Figure 9.10 Example internetwork topology.

bit rate the lower the cost value. As we shall see, the cost value associated with each line is used during the routing of datagrams/packets and hence is also known as a **routing metric**. The cost of a route/path through the interconnection network is determined by summing together the cost value associated with each line that makes up the path. This is known as the **path cost** and, when different paths between two routers are available, the path with the least path cost value is known as the **shortest path**. Other metrics used in relation to the computation of the shortest paths are based on the physical length – and hence propagation delay – of each line, the number of lines (**hops**) in the route (**hop count**), and the mean queuing delay within each router associated with a line.

Let us assume that host A on netid 1 wants to send a datagram to host B on netid 3. As we saw in Section 9.5.1, on determining that the destination netid in the datagram header is for a different netid from its own, gateway G1 forwards the datagram to router R1 over the connecting line/link. On receipt of the datagram, R1 proceeds to forward it first to R3 over the interconnection network and then to G3. At this point, the IP in G3 knows how to route the datagram to host B using the hostid part of the destination IP address and the related MAC address of B in its ARP cache. What we do not know, is how the datagram is routed across the interconnection network.

There are three unanswered questions involved:

(1) How does R1 know from the netid contained within the destination IP address that the destination router is R3?

(2) How does R1 know the shortest path route to be followed through the interconnection network to R3?

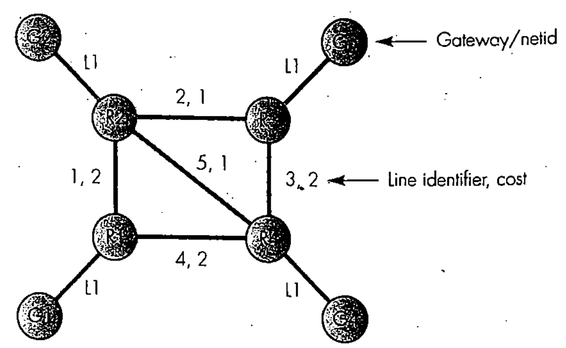
(3) How does R3 know how to relay the datagram to G3 instead of one of the other gateways that is attached to it?

In relation to the last point, we can accept that a simple protocol can be used to enable each gateway to inform the router to which it is attached, the netid of the access network. The first two points, however, are both parts o the routing algorithm associated with the interconnection network. There are a number of different algorithms that can be used and we shall describe a selection of them in the following subsections. Note that when discussin routing algorithms, the more general term "packet" is used.

9.6.1 Static routing

With this type of routing, the outgoing line to be used to reach all netids is loaded into the routing table of each router when it is first brought into service. As an example, we show the routing table for each of the four routers in the example internet in Figure 9.11(b). To avoid unnecessary repetition, we assume only one gateway/netid is attached to each router.

(a)



(b)

R1:

Netid	Line
1	L1
2	1
3	1
4	4

R2:

Netid	Line
1	1
2	L1
3	2
4	5

R3:

Netid	Line
1	2
2	2
3	L1
4	3, 2

R4:

Netid	Line
1	4
2	5
3	3, 5
4	L1

Figure 9.11 Static routing: (a) internet topology; (b) routing table entries.

To route a packet from a host attached to, say, netid 1 to another that is attached to netid 3, on receipt of the packet, R1 consults its routing table and determines it should forward the packet on line 1. Similarly, on receipt of the packet, R2 determines it should be forwarded on line 2. Finally, R3 forwards the packet to G3 and from there it is forwarded by G3 to the host specified in the hostid part of the IP address.

As we show in the routing tables for routers R3 and R4, two alternative lines are given for routing packets between these two routers. As we can deduce from the cost values, both routes have the same path cost of 2, one using only line 3 and the other going via R2 and lines 2 and 5. Clearly, however, if a second routing metric of, say, distance was used, then the second path would be longer and hence only a single path would be present. For this reason, more than one metric is sometimes used and the choice of path is then based on the information contained within the related set of routing tables.

We can also make a second observation from this set of routing tables; that is, to go from R1 to R3, the shortest path is via R2 using lines 1 and 2. Also, when we look at the routing table for R2, the shortest path from R2 to R3 is also line 2. More generally, if the shortest path between two routers, A and C, is via an intermediate router B, then the shortest path from B to C is along the same path. This is known as the **optimality principle** and it follows from this that each router along the shortest path need only know the identity of its immediate neighbor along the path. The routing operation is known, therefore, as **next-hop routing** or **hop-by-hop routing**.

The disadvantage of static routing is that all the routing table entries may need to be changed whenever a line is upgraded or a new line is added. Also, should a line or router develop a fault, when the fault is reported, the routing tables in all affected routers need to be changed. For these reasons, static routing is inappropriate for a large, continuously-changing network like the Internet.

9.6.2 Flooding

To explain the operation of the flooding algorithm, we return to Figure 9.10 and again assume we are sending a datagram from host A to host B. The steps followed are summarized in Figure 9.12.

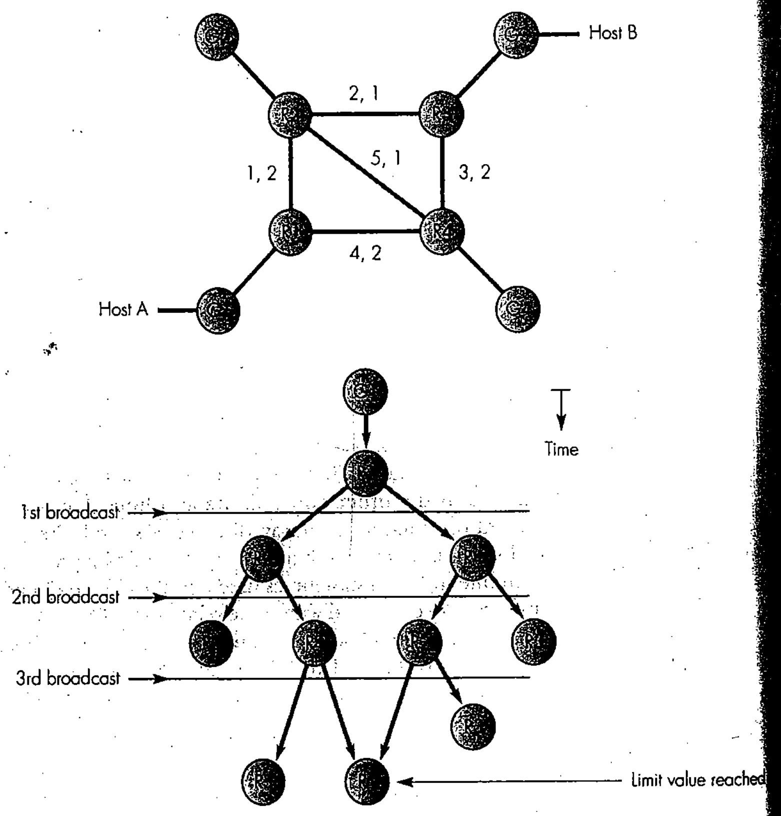


Figure 9.12 Flooding example.

On receipt of the packet from G1, R1 sends a copy of it over both lines 1 and 4. Similarly, on receipt of their copy of the packet, routers R2 and R4 determine from the netid within it that the packet is not for one of their own networks and hence proceed to forward a copy of the packet onto lines 2 and 5 (R2) and lines 5 and 3 (R4). However, since line 2 has a higher bit rate – lower cost value – than line 3, the copy of the packet from R2 will arrive at R3 first and, on determining it is addressed to one of its own netids, R3 forwards the packet to G3. Additional copies of the packet will then be received by R3 but, remembering that each copy will have the same value in the *identifier* field, these can be detected as duplicates and discarded by R3.

In order to limit the number of copies of the packet that are produced, a limit is set on the number of times each copy of the packet is forwarded. In the example, it is assumed that a limit value of 3 has been placed in the *time-to-live* field of the packet header by R1. Prior to forwarding copies of the packet, the limit value is decremented by 1 by the recipient router and only if this is above zero are further copies forwarded.

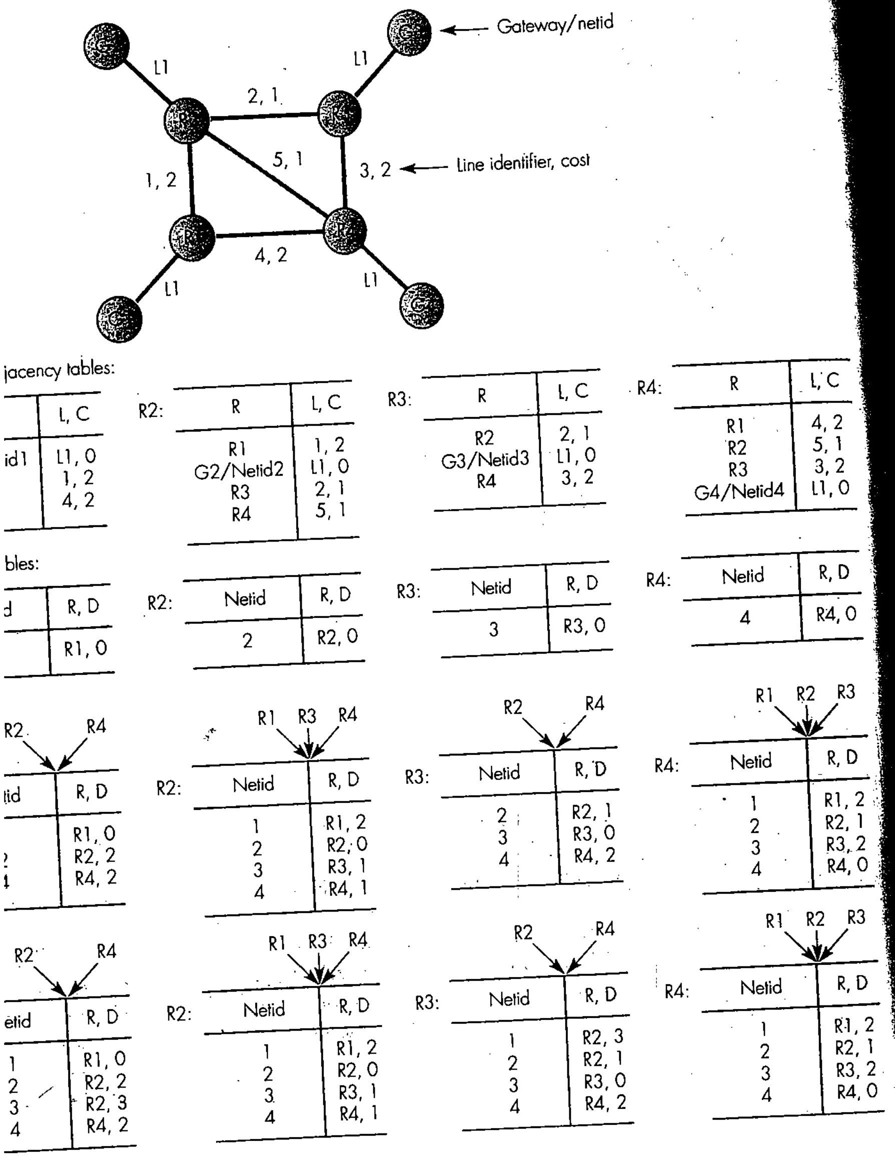
As we can deduce from the figure, the flooding algorithm ensures that the first copy of the packet flows along the shortest path and hence is received in the shortest time. Also, should a line or router fail, providing an alternative path is available, a copy of each packet should always be received. Flooding, therefore, is an example of an **adaptive** – also known as **dynamic** – routing algorithm. Nevertheless, as we can see from this simple example, even with a limit of three hops, the packet is transmitted 10 times. This compares with just two transmissions using the shortest path. Hence the very heavy bandwidth overheads associated with flooding means that it is inappropriate for use with the large networks that make up the Internet.

9.6.3 Distance vector routing

The distance vector algorithm is a distributed algorithm that enables each router to build up a routing table (the vector) that contains the path cost (the distance) to reach all the netids in the internetwork.

Initially, each router knows only the identity of, firstly, the netids of the networks that are attached to it – through gateways – and their related local line numbers, and secondly, the identity of the lines – and their cost – that form direct links to other routers. Normally, this information is either entered by network management or by the exchange of configuration messages with the other routers when each router is first brought into service. The information is held in a table known as a *connectivity* or *adjacency* table and the contents of the four tables for our example internetwork, together with the contents of the initial routing table for each router, are shown in Figure 9.13(a).

In order for each router to build up its complete routing table – containing the minimum distance (shortest path) to reach all netids – at predefined time intervals, each router first adds the known cost of the lines that connect



jacency tables:

	L, C
id1	L1, 0 1, 2 4, 2

R2:

R	L, C
R1 G2/Netid2 R3 R4	1, 2 L1, 0 2, 1 5, 1

R3:

R	L, C
R2 G3/Netid3 R4	2, 1 L1, 0 3, 2

R4:

R	L, C
R1 R2 R3 G4/Netid4	4, 2 5, 1 3, 2 L1, 0

bles:

d	R, D
	R1, 0

R2:

Netid	R, D
2	R2, 0

R3:

Netid	R, D
3	R3, 0

R4:

Netid	R, D
4	R4, 0

R2 R4

tid	R, D
	R1, 0 R2, 2 R4, 2

R1 R3 R4

R2:

Netid	R, D
1 2 3 4	R1, 2 R2, 0 R3, 1 R4, 1

R2 R4

R3:

Netid	R, D
2 3 4	R2, 1 R3, 0 R4, 2

R1 R2 R3

R4:

Netid	R, D
1 2 3 4	R1, 2 R2, 1 R3, 2 R4, 0

R2 R4

etid	R, D
1 2 3 4	R1, 0 R2, 2 R2, 3 R4, 2

R1 R3 R4

R2:

Netid	R, D
1 2 3 4	R1, 2 R2, 0 R3, 1 R4, 1

R2 R4

R3:

Netid	R, D
1 2 3 4	R2, 3 R2, 1 R3, 0 R4, 2

R1 R2 R3

R4:

Netid	R, D
1 2 3 4	R1, 2 R2, 1 R3, 2 R4, 0

.13 Distance vector algorithm: (a) internet topology and initial tables; (b) derivation outing tables.

the router to its neighbors to the current distance values in its own routing table and forwards a copy of the related updated table to each of its neighbors. Then, based on the information received, if a reported distance is less than a current entry, each router proceeds to update its own routing table with the reported distance. The same procedure then repeats with the updated table contents. This procedure repeats for a defined number of iterations after which, each router has determined the path with the minimum distance to be followed to reach all netids.

As an example, the build-up of the final routing table for each of the four routers in our example internetwork is shown in Figure 9.13(b). To avoid repetition, we assume that only a single gateway/netid is attached to each router and, as we can see, for this simple internet the contents of each routing table are complete after just two routing table updates.

In the case of R1, this receives the updated contents of the routing tables held by R2 and R4. Hence after R1 receives the first set of updated tables from them, it determines that the shortest path to reach netid 2 has a distance of 2 via R2 and, to reach netid 4, the distance is 2 via R4. At the same time, R2 and R4 have themselves received update information from their own neighbors and, as a result, on receipt of the second set of updated tables from them, R1 determines that the shortest path to reach netid 3 has a distance of 3 via R2. Note that with the distance vector algorithm an entry is updated only if a new distance value is less than the current value. Also, that routes with equal path cost values are discarded.

The final routing table of each router contains the next-hop router and the corresponding distance (path cost) value to reach all of the netids in the internetwork. Hence to route a packet, the netid is first obtained from the destination IP address in the packet header and the identity of the next-hop router read from the routing table. The corresponding line number on which the packet is forwarded is then obtained from the connectivity table.

To ensure that each table entry reflects the current active topology of the internet, each entry has an associated timer and, if an entry is not confirmed within a defined time, then it is timed-out. This means that each router transmits the contents of its complete routing table at regular intervals which, typically, is every 30 seconds. Again, for a small internet this is not a problem but for a large internet like the Internet, the bandwidth and processing overheads associated with the distance vector algorithm can become very high. Also, since entries are updated in the order in which they are received and paths of equal distance/cost are discarded, routers may have dissimilar routes to the same destination. As a result, packets addressed to certain destinations may loop rather than going directly to the desired router/gateway. Nevertheless, the **routing information protocol (RIP)** which uses the distance vector routing algorithm is still widely used in many of the individual networks that make up the Internet.

.6.4 Link-state shortest-path-first routing

As the name implies, this type of routing is based on two algorithms: link-state (LS) and shortest-path-first (SPF). The link-state algorithm is used to enable each router to determine the current (active) topology of the internet and the cost associated with each line/link. Then, once the topology is known, each router runs (independently) the shortest-path-first algorithm to determine the shortest path from itself to all the other routers in the internet.

Link-state algorithm

As with the distance vector algorithm, initially, each router knows only its own connectivity/adjacency information and, as an example, the table entries for our example internet are repeated in Figure 9.14(a). The link-state algorithm is then run and the build-up of the internet topology by R1 is shown in Figure 9.14(b).

Initially, based on the information R1 has in its own connectivity table, the (incomplete) topology is as shown in (i). At regular intervals, each router broadcasts a **link-state message**, containing the router's identity and its associated connectivity information, to each of its immediate neighbors. Hence in the example, we assume that R2 is the first to send its own connectivity information to R1 and this enables R1 to expand its knowledge of the topology to that shown in (ii). This is followed by the connectivity information of R4 which enables R1 to expand its knowledge of the topology to that shown in (iii). Concurrently with this happening, the same procedure will have been carried out by all of the other routers. Hence in our example internet, R2 and R4 will have received the connectivity information of R3. After this has been received, therefore, R2 and R4 relay this information on to R1 in a second set of link-state messages and this enables R1 to complete the picture of the active topology (iv). Also, since each router has carried out the same procedure, each will have derived the current active topology and, in addition, determined the identity of the router to which each netid is attached. At this point, each router runs the shortest-path-first algorithm to determine the shortest path from itself to all the other routers. In practice, there are a number of algorithms that can be used to find the shortest path but we shall restrict our discussion to the Dijkstra algorithm.

Dijkstra shortest-path-first algorithm

We shall explain the Dijkstra algorithm in relation to our example internet topology. This is shown in Figure 9.15(a) together with the cost of the lines that link the routers together. The sequence of steps followed by R1 to derive the shortest paths to reach the other three routers is shown in Figure 9.15(b).

Shown in parentheses alongside each of the other routers is the aggregate cost from that router back to the source via the router indicated. Hence an entry of (4,R4) means that the cost of the path back to R1 is 4 via R4. Initially, only the path cost of those routers that are directly connected to R1 are known (R2 and R4) and those not directly connected (R3) are marked

(a)
Connectivity/adjacency tables:

R1:

R	L, C
G1/Netid1	L1, 0
R2	1, 2
R4	4, 2

R2:

R	L, C
R1	1, 2
G2/Netid2	L1, 0
R3	2, 1
R4	5, 1

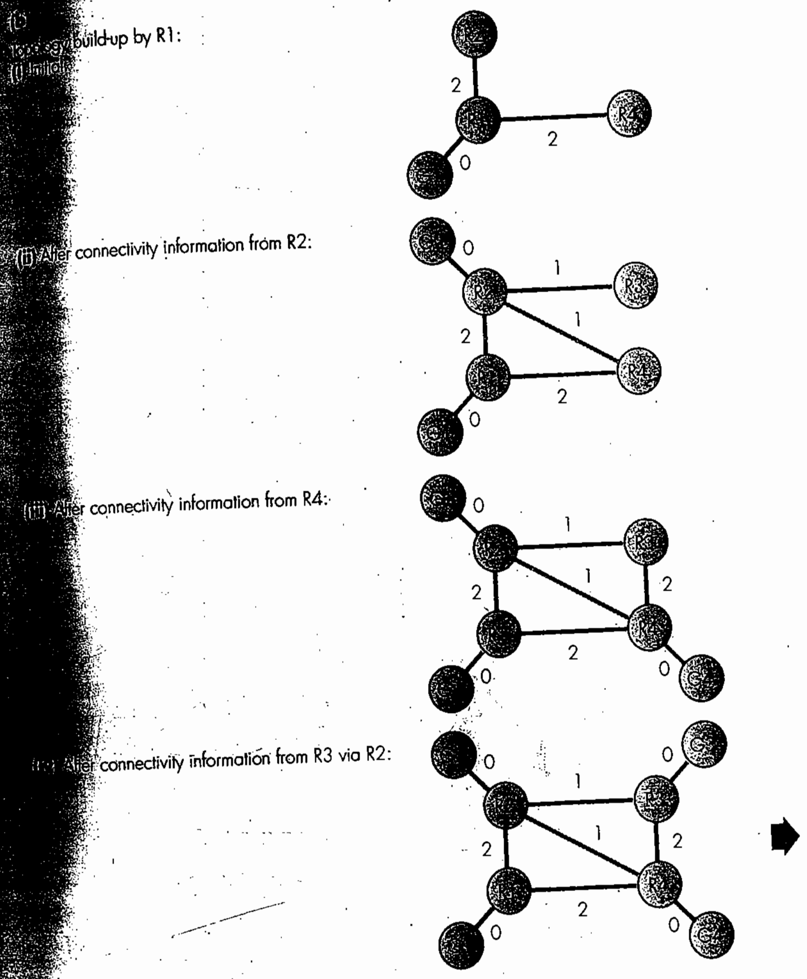
R3:

R	L, C
R2	2, 1
G3/Netid3	L1, 0
R4	3, 2

R4:

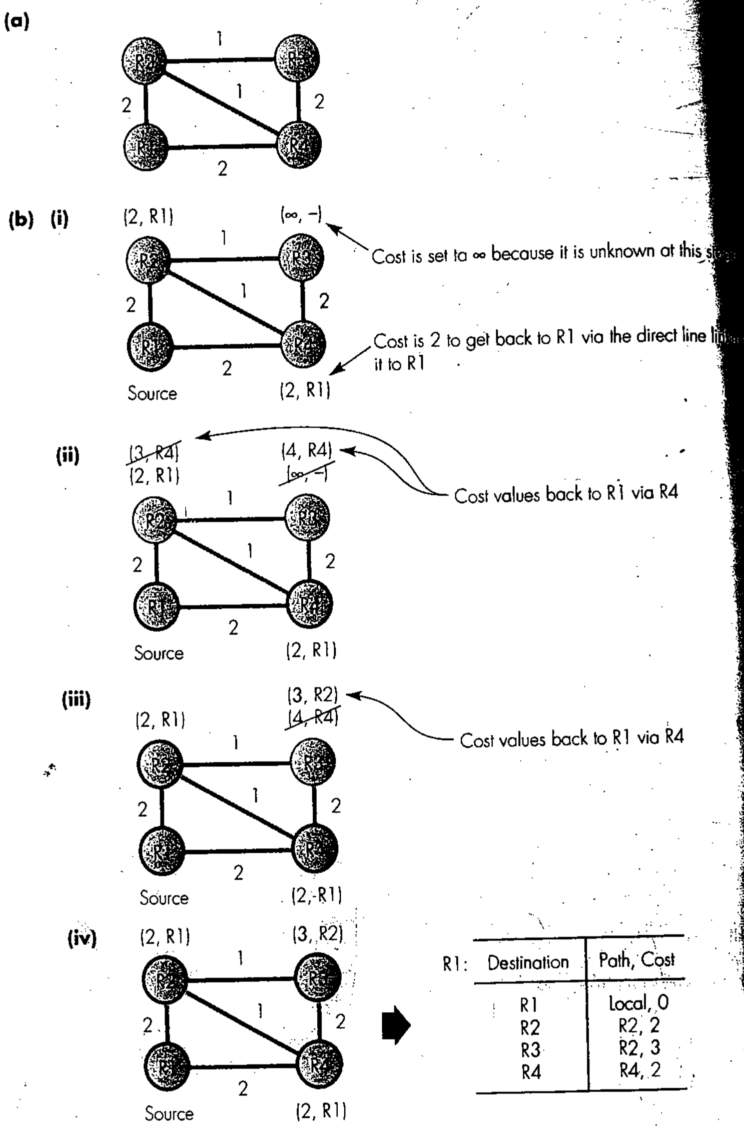
R	L, C
R1	4, 2
R2	5, 1
R3	3, 2
G4/Netid4	L1, 0

(b)
Topology build-up by R1:


(i) Initial:
R2
2
R1
R4
2
G1
0
(ii) After connectivity information from R2:
G2
0
1
R2
R3
1
2
R1
R4
2
G1
0
(iii) After connectivity information from R4:
G2
0
1
R2
R3
1
2
2
R1
R4
2
G1
0
0
G4
(iv) After connectivity information from R3 via R2:
G2
0
G3
0
1
R2
R3
1
2
2
R1
R4
2
G1
0
0
G4

G/Netid	R
G1/1	R1
G2/2	R2
G3/3	R3
G4/4	R4

Figure 9.14 Link state algorithm: (a) initial connectivity/adjacency tables; (b) derivation of active topology and netid location.



Destination	Path, Cost
R1	Local, 0
R2	R2, 2
R3	R2, 3
R4	R4, 2

Figure 9.15 Dijkstra algorithm: (a) initial topology; (b) derivation of shortest paths from R1 to each other router.

with an infinite path cost value. Also, until a cost value is known to be the minimum cost, it is said to be **tentative** and only when the cost value is confirmed as the minimum value is it said to be **permanent**. The router is then shown in bold.

Initially, since R1 is the source it is shown in bold and the path costs back to R1 from the two directly connected routers (R2 and R4) are shown equal to the respective line costs (i). Hence R2, for example, has an entry of (2,R1) indicating the cost is 2 to get back to R1 via the direct line linking it to R1. Also, since R3 is not connected directly to R1, it is shown with a path cost of infinity.

Once this has been done, the next step (ii) is to choose the router with the minimum path cost value from all the remaining routers that are still tentative. Hence in our example, the choice is between R2 and R4 – since both are tentative and have a path cost value of 2 – and, arbitrarily, we have chosen R4. This is now marked permanent and the new set of aggregate path cost values via R4 are computed. For example, the cost of the path from R2 to R1 via R4 is 3 (1 from R2 to R4 plus 2 from R4 to R1) but, since this is greater than the current cost of 2, this is ignored. In the case of R3, however, the cost of 4 via R4 is less than the current value of infinity and hence (4,R4) replaces the current entry.

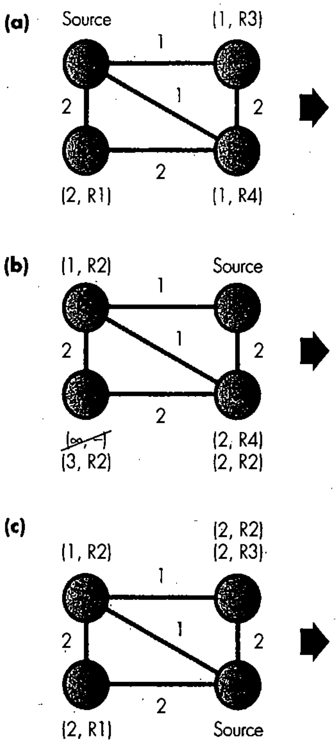
The router with the minimum path cost value is again chosen from those that remain tentative and, since R2 has a path cost of 2, this is marked permanent and the new path costs to R1 via R2 are computed (iii). As we can see, the path cost from R3 to R1 via R2 is only 3 and hence an entry of (3,R2) replaces the current entry of (4,R4). Finally (iv), R3 is made permanent as it is the only remaining router that is still tentative and, now that the minimum path costs from each of the other routers back to R1 are known, the routing table for R1 is complete.

In Figure 9.16 we show the same procedure applied first with R2 as the source – part (a) – then with R3 – part (b) – and finally with R4 – part (c). From these derivations we can make some observations about the algorithm:

- The derived shortest path routes adhere to the optimality principle.
- If the computed path costs associated with two or more tentative routers are the same, then an arbitrary selection can be made as to which is made permanent.
- If the computed aggregate path cost from a (tentative) router to the source via a different router is the same as that via another router, then both can be retained. The choice of route is then arbitrary and load sharing becomes possible.

Datagram routing procedures

The routing of a datagram involves a combination of both link-state tables – one containing the location of all netids and the other the connectivity information – and the derived set of shortest-path routing tables. These are used in slightly different ways depending on the choice of routing method,


(a)
Source
(1, R3)
1
2
1
2
2
(2, R1)
(1, R4)
(b)
(1, R2)
Source
1
2
1
2
2
(3, R2)
(2, R4)
(2, R2)
(c)
(2, R2)
(1, R2)
(2, R3)
1
2
1
2
2
(2, R1)
Source

R2:

Destination	Path, Cost
R1	R1, 2
R2	Local, 0
R3	R3, 1
R4	R4, 1

R3:

Destination	Path, Cost
R1	R2, 3
R2	R2, 1
R3	Local, 0
R4	R4/R2, 2

R4:

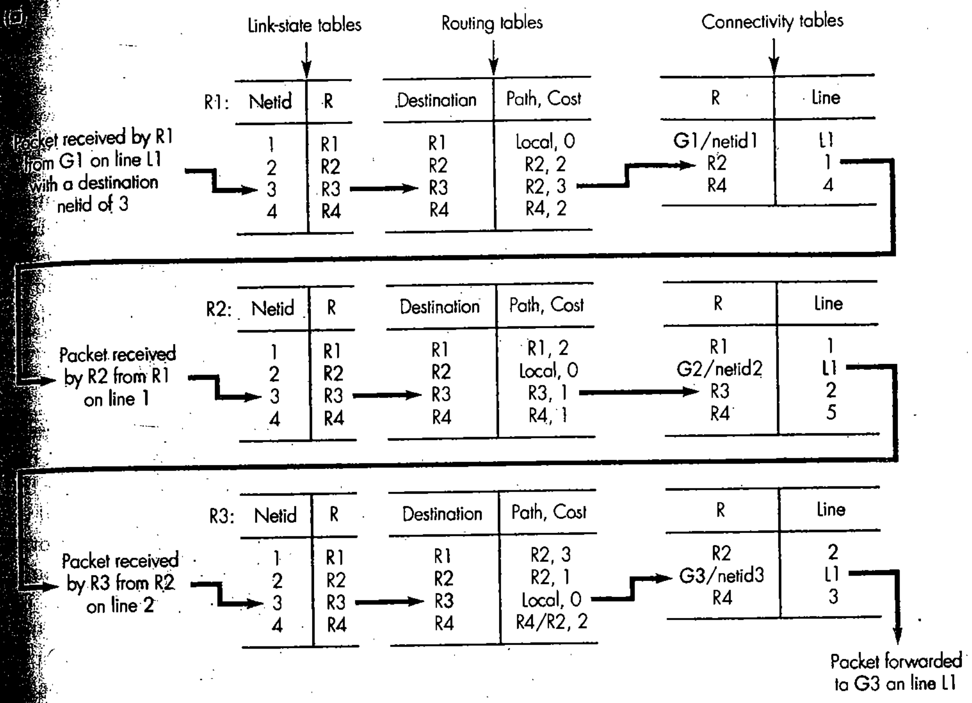
Destination	Path, Cost
R1	R1, 2
R2	R2, 1
R3	R3/R2, 2
R4	Local, 0

Figure 9.16 Shortest path derivations: (a) by R2; (b) by R3; (c) by R4.

hop-by-hop routing or source routing. We shall explain the procedure followed with each method using the example of a host attached to netid 1 sending a datagram/packet to a host attached to netid 3.

The procedure followed with hop-by-hop routing is summarized in Figure 9.17(a). Using this method, each router computes only its own routing table contents and uses this together with the contents of its own connectivity table. On receipt of the packet from gateway G1, router R1 obtains the netid from the destination IP address in the packet header – netid 3 – and uses its copy of the link-state table to determine that this is reached via router R3. It then determines from the contents of its routing table that the next-hop router on the shortest path to R3 is R2 and hence proceeds to forward the packet to R2 over the line indicated in its connectivity table, line 1.

The same procedure is repeated by router R2 – using its own link-state, routing, and connectivity tables – to forward the packet to R3 over line 2. Finally, on receipt of the packet, R3 determines from its own tables that the packet is addressed to netid 3 and that this is attached to one of its local lines, L1. The packet is then forwarded to the attached gateway and from there to the destination host.


(a)
Link-state tables
Routing tables
Connectivity tables
Packet received by R1 from G1 on line L1 with a destination netid of 3
Packet received by R2 from R1 on line 1
Packet received by R3 from R2 on line 2
Packet forwarded to G3 on line L1

R1:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	Local, 0
R2	R2, 2
R3	R2, 3
R4	R4, 2

R	Line
G1/netid1	L1
R2	1
R4	4

R2:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	R1, 2
R2	Local, 0
R3	R3, 1
R4	R4, 1

R	Line
R1	1
G2/netid2	L1
R3	2
R4	5

R3:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	R2, 3
R2	R2, 1
R3	Local, 0
R4	R4/R2, 2

R	Line
R2	2
G3/netid3	L1
R4	3

(b)

1. Datagram received by R1 from G1 on line L1
2. R1 uses its set of routing tables to determine the least path cost is via routers R2, R3, and writes this list into an options field in the datagram header.
3. R1 uses its own connectivity table to forward the packet to the first router in the list, R2, using line 1.

↓

4. On receipt of the packet, R2 reads the second router from the list, R3, and uses its own connectivity table to forward the packet on line 2.

↓

5. On receipt of the packet, R3 determines it is for one of its local gateways and uses its own connectivity table to forward the packet to G3 on line L1.

Figure 9.17 LS-SPF routing examples: (a) hop-by-hop routing; (b) source routing.

The procedure followed with source routing is summarized in Figure 9.17(b). Using this method, once all the routers have built up a picture of the current active topology using the link-state algorithm, they each compute the complete set of four routing tables. Then, on receipt of a packe from one of its attached gateways – G1 in the example – the source router R1 – uses the set of tables to determine the list of routers that form the shortest path to the intended destination – R2 and R3. The list is then inserte into an *options* field of the datagram header by R1 and the packet forwarde to the first router in the path, R2, using the corresponding line numbe obtained from R1's own connectivity table, line 1.

On receipt of the packet, R2 reads from the *options* field the identity of the next router along the path, R3, and uses its own connectivity table to determine the line the packet should be forwarded on, line 2. On receipt of the packet, R3 determines it is intended for one of its local gateways and uses its own connectivity table to determine the packet should be forwarded to gateway G3 on line L1.

Additional comments

Although in the various examples, internet-wide identifiers have been used to identify each of the lines in the example internet topology, this has been done to simplify the related descriptions. In practice, as we can deduce from the description of the LS-SPF algorithm the line identifiers associated with each router have only local significance and, since these are part of the router's configuration information, normally, a different set of line identifiers is used by each router.

In our discussion of the link-state algorithm, we assumed that the transmission of the link-state messages was reliable and that none was lost as a result of transmission errors. Clearly, should a link-state message be corrupted, then the routing tables in each router may be inconsistent and, amongst other things, cause packets to loop. To overcome this, each link-state message, in addition to the identity of the router that created the message and its associated connectivity information, also contains a sequence number and a timeout value. As we have mentioned, link-state information is distributed by each router relaying a copy of the messages it receives from each of its neighbors on to its other neighbors. Hence to avoid messages being relayed unnecessarily, when each new message it created – at defined tim intervals – it is assigned a sequence number equal to the previous numbe plus one. Each router then keeps a record of the sequence number contained within the last message it received from each of the other routers and only if a new message is received – that is, one with a higher sequence number – is a copy forwarded to its other neighbors. In addition, the associated timeout value in each message is decremented by each router and, should this reach zero, the received message is discarded.

Although in the simple example we used to describe the LS-SPF algorithm only a single routing metric (cost value) was used, multiple metrics can be used. In such cases, therefore, there may be a different shortest path between each pair of routers for each metric. Although this leads to

additional computation overheads, the choice of path can then be made dependent on the type of information contained within the datagram being routed. For example, for real-time information such as digitized speech, the choice of path may be based on minimum delay rather than bit rate.

As we can deduce from the description of the distance vector algorithm in Section 9.6.3, for large internets, the amount of routing information passed between routers is substantial and, in the limit, involves each router transferring the contents of its complete routing table at regular intervals. In contrast, the link-state shortest-path-first algorithm involves only the transfer of the link-state information of each router. Hence it is far more efficient in terms of the amount of bandwidth that is utilized for routing updates. It is for this reason, that the LS-SPF algorithm is now the preferred algorithm. The protocol based on this is known as the open shortest path first (OSPF) routing protocol and, as we showed earlier in Figure 9.2, this forms an integral part of the IP.

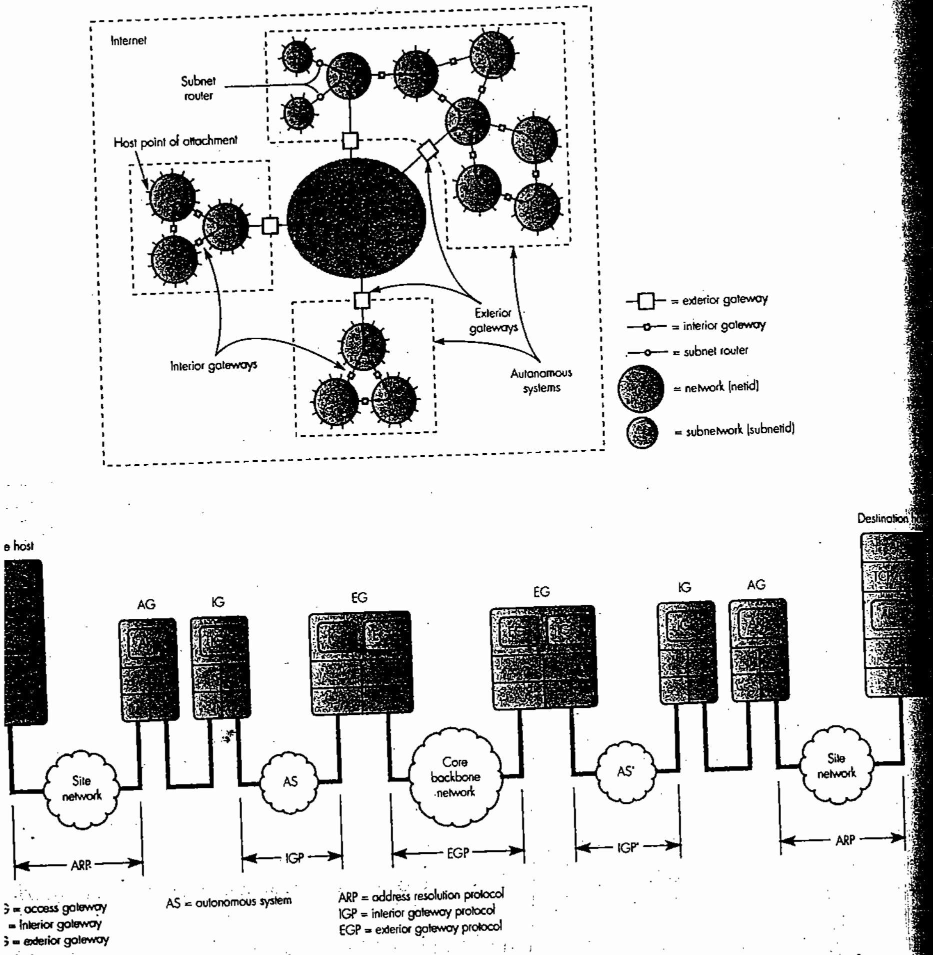
9.6.5 Hierarchical routing

As we showed earlier in Figure 9.9 and explained in the accompanying text, the Internet is made up of a large number of separately managed networks/internetworks that are interconnected together by means of a global backbone network. Each separate network/internetwork is treated as an **autonomous system (AS)** – with its own internal routing algorithm and management authority – and is assigned a number to identify it within the context of the total Internet. The general architecture is shown in Figure 9.18(a) together with some (very much simplified) autonomous system topologies.

To discriminate between the routers used within the various types of network, the following terminology is used:

- **subnet router:** this is a router that operates entirely within a single network when subnetting is being used;
- **access gateway:** this is used to connect an access network to an interior gateway;
- **interior gateway:** this is a router that is used to interconnect the networks within an autonomous system (typically a regional or national network);
- **exterior gateway:** this is a router that is used to connect an autonomous system to the core backbone network.

The various routing protocols associated with the Internet are identified in Figure 9.18(b). The routing protocol used by the interior gateways within a single autonomous system is known as an **interior gateway protocol (IGP)** and that used by the exterior gateways within the backbone network an **exterior gateway protocol (EGP).** Since the Internet consists of an interconnected


Internet
Subnet router
Host point of attachment
Exterior gateways
Interior gateways
Autonomous systems
= exterior gateway
= interior gateway
= subnet router
= network (netid)
= subnetwork (subnetid)
e host
Destination
AG
IG
EG
EG
IG
AG
Site network
AS
Core backbone network
AS'
Site network
ARP
IGP
EGP
IGP'
ARP
= access gateway
= interior gateway
= exterior gateway
AS = autonomous system
ARP = address resolution protocol
IGP = interior gateway protocol
EGP = exterior gateway protocol

ure 9.18 Hierarchical routing over the Internet: (a) generalized architecture and minology; (b) associated routing protocol terminology.

set of networks and internetworks, many of which have evolved over a r
tively long period of time, each autonomous system may have a differ
IGP. Examples are the RIP and the OSPF the principles of which
described in the previous two sections. The EGP, however, as it must be, is
Internet-wide standard.

To reflect the different types of routing protocols involved, the total routing information is organized hierarchically:

- Hosts maintain sufficient routing information to enable them to forward packets directly to other hosts on the same network, a subnet router (if subnetting is being used), and to an access gateway.
- Subnet routers maintain sufficient routing information to enable them to forward packets to other subnet routers (belonging to the same network) or to the network access gateway.
- Interior gateways maintain sufficient routing information to forward packets to their own access gateways and to other interior and exterior gateways within the same autonomous system.
- Exterior gateways maintain sufficient routing information to forward packets to an interior gateway (if the packet is for the autonomous system to which it is attached) or to another exterior gateway if it is not.

An example protocol used by hosts to route packets to other hosts (and an access gateway) that are attached to the same network is the ARP we described earlier in Section 9.5. Subnet routers are used when a network is divided up into subnets to interconnect the various subnets together. Hence they operate in a similar way to interior gateways – and use similar routing protocols – except that all routing within the network is carried out using the subnet address within the hostid part of the destination IP address rather than the netid part. As we indicated earlier, the interior gateway routing protocol can vary from one autonomous system to another. Example protocols are the RIP (based on the distance vector algorithm which we described in Section 9.6.3) and the OSPF (based on the link-state shortest-path-first (LS-SPF) algorithm we described in Section 9.6.4). In this section, we shall describe the standard exterior gateway protocol known as the **border gateway protocol (BGP)**.

Border gateway protocol

The management authority associated with each autonomous system nominates one or more gateways to function as exterior gateways for that system. Within the autonomous system, these communicate with the other interior gateways using the interior gateway protocol for that system. Each exterior gateway, through its local routing table, knows about the netids within that system and their distances from that gateway. The contents of the routing table are built up in the same way as the interior gateways of the autonomous system to which it is attached.

When each exterior gateway is first initialized, it is given the unique identity of the autonomous system to which it is attached and the contents of a connectivity table. This is known as the **reachability table** and enables the gateway to communicate with all the other exterior gateways to which it has a direct link. The BGP in each exterior gateway then makes contact with the

BGP in other selected exterior gateways to exchange routing information with them. This routing information consists of the list of netids within the corresponding autonomous system together with their distances and routes from the reporting exterior gateway. This information is used by a sending gateway to select the best exterior gateway for forwarding datagrams to a particular netid and hence autonomous system.

The three main functions associated with the BGP are as follows:

- neighbor acquisition and termination,
- neighbor reachability,
- routing update.

Each function operates using a request-response message exchange. The messages associated with each function are shown in Table 9.1

Since each autonomous system is managed and run by a different authority, before any routing information is exchanged, two exterior gateways attached to different systems must first agree to exchange such information. This is the role of the **neighbor acquisition and termination** procedure. When two gateways agree to such an exchange, they are said to have become **neighbors.** When a gateway first wants to exchange routing information, it sends an *acquisition request* message to the BGP in the appropriate gateway which then returns either an *acquisition confirm* message or, if it does not want to accept the request, an *acquisition refuse* message which includes a reason code.

Table 9.1 BGP message types that are exchanged by the BGP in exterior gateways and their meaning

Function	*BGP message*	*Meaning*
Neighbor acquisition and termination	Acquisition request	Requests [illegible] neighbor
	Acquisition confirm	Gateway agrees [illegible]
	Acquisition refuse	Gateway refuses [illegible]
	Cease request	Requests [illegible] relationship
	Cease confirm	Confirms [illegible]
Neighbor reachability	Hello	Requests neighbor [illegible] previously [illegible]
	I-heard-you	Confirm relationship
Routing update	Poll request	Requests network [illegible]
	Routing update	Network reachability [illegible]
Error response	Error	Response to any [illegible] message

Once a neighbor relationship has been established between two gateways – and hence autonomous systems – they periodically confirm their relationship. This is done either by exchanging specific messages – *hello* and *I-heard-you* – or by embedding confirmation into the header of normal routing information messages.

The actual exchange of routing information is carried out by one of the gateways, which sends a *poll request* message to the other gateway asking it for the list of networks (netids) that are reachable via that gateway and their distances from it. The response is a *routing update* message which contains the requested information. Finally, if any request message is incorrect, an *error message* is returned as a response with an appropriate reason code.

As with the other IP protocols, all the messages (PDUs) associated with the BGP are carried in the user data field of an IP datagram. All BGP messages have the same fixed header, the format of which is shown in Figure 9.19.

The *version* field defines the version number of the BGP. The *type* and *code* fields collectively define the type of message while the *status* field contains message-dependent status information. The *checksum*, which is used as a safeguard against the processing of erroneous messages, is the same as that used with IP. The *autonomous system number* is the assigned number of the autonomous system to which the sending gateway is attached; the *sequence number* is used to synchronize responses to their corresponding request message.

Neighbor reachability messages contain only a header with a *type* field of 5, a code of 0 = hello, and a 1 = I-heard-you.

Neighbor acquisition messages have a *type* field of 3; the code number defines the specific message type. The *hello interval* specifies the frequency with which hello messages should be sent; the *poll interval* performs the same function for poll messages.

A poll message has a *type* field of 2. The *code* field is used to piggyback the neighbor reachability information: a code of 0 = hello and a code of 1 = I-heard-you. The *source network IP address* in both the poll and the routing update response messages indicates the network linking the two exterior gateways. This allows the core network itself to consist of multiple networks with an associated routing protocol.

The *routing update* message contains the list of networks (netids) that are reachable via each gateway within the autonomous system arranged in distance order from the responding exterior gateway. As indicated, this enables the requesting gateway to select the best exterior gateway through which to send a packet for forwarding within an autonomous system. Note that to conserve space, each netid address is sent in three bytes (24 bits) only with the most significant 8-bit hostid field missing. The latter is redundant for all IP address class types.

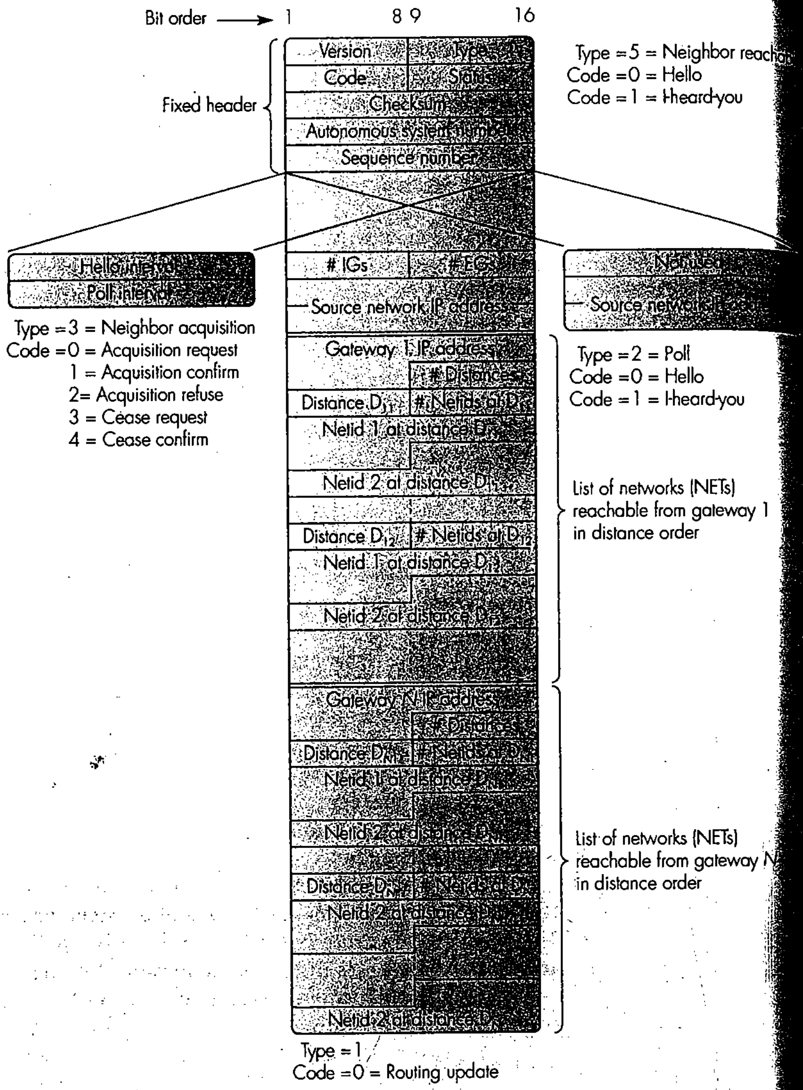

Bit order → 1 8 9 16
Version
Type
Code
Status
Checksum
Autonomous system number
Sequence number
Fixed header
Type = 5 = Neighbor reach
Code = 0 = Hello
Code = 1 = I-heard-you
Hello interval
Poll interval
Type = 3 = Neighbor acquisition
Code = 0 = Acquisition request
1 = Acquisition confirm
2 = Acquisition refuse
3 = Cease request
4 = Cease confirm
# IGs
# EGs
Source network IP address
Netid
Source network IP add
Gateway 1 IP address
# Distances
Distance D11
# Netids at D11
Netid 1 at distance D11
Netid 2 at distance D11
Distance D12
# Netids at D12
Netid 1 at distance D12
Netid 2 at distance D12
Type = 2 = Poll
Code = 0 = Hello
Code = 1 = I-heard-you
List of networks (NETs) reachable from gateway 1 in distance order
Gateway N IP address
# Distances
Distance DN1
# Netids at DN1
Netid 1 at distance DN1
Netid 2 at distance DN1
Distance DN2
# Netids at DN2
Netid 2 at distance DN2
List of networks (NETs) reachable from gateway N in distance order
Type = 1
Code = 0 = Routing update

Figure 9.19 Border gateway protocol (BGP) message formats.

9.6.6 Classless inter-domain routing

As we explained earlier in Section 9.4, each IP address is 32 bits in length and is made up of a netid part and a hostid part. The allocation of netids is centrally managed by the Internet Network Information Center and, in order to utilize network addresses efficiently, the number of bits used for the netid part varies. As we showed earlier in Figure 9.5, the number of bits is determined by the address class:

Class A: netid = 7 bits, hostid = 24 bits
Class B: netid = 14 bits, hostid = 16 bits
Class C: netid = 21 bits, hostid = 8 bits

In this way, the manager of a network with many attached hosts can be allocated a class A address, one with a small number of hosts a class C address, and the rest a class B address.

In practice, however, when requesting an IP address, most network managers opted for a class B address since they considered a class C address with just 256 hostids too small and a class A address with 2^{24} hostids too large. As a result, even though there were plenty of netids available with class C addresses, a shortage of class B addresses was predicted. To resolve this problem, an alternative type of routing (which essentially bypasses the fixed divisions associated with each class) known as **classless inter-domain routing (CIDR)** was introduced. It is defined in **RFC 1519**.

As we saw in Example 9.1, there are over 2 million class C addresses and hence the primary aim of CIDR was to exploit their usage in a more efficient way. To do this, with CIDR, instead of using the fixed netid/hostid boundary associated with class C addresses, the boundary is made variable and dependent on the number of attached hosts specified by the network manager making a request. For example, if the manager of a new network to be connected to the Internet estimates that the number of hosts may grow to, say, 1000, then a contiguous block of 1024 (2^{10}) class C addresses is allocated.

The alternative CIDR method was introduced relatively recently (post-1996) and, by that time, the general Internet architecture had grown to that we showed earlier in Figure 9.9. Hence in order to reduce the amount of routing information exchanged by exterior gateways, it was decided to abandon the fixed netid/hostid boundary and instead introduce a hierarchical structure that reflected this architecture. Remembering that all class C addresses start with 110, the range of addresses available (in dotted decimal form) is from:

192.0.0.0 through to 223.255.255.255

Clearly, some of these addresses had already been allocated. Hence allocations to the networks in the different continents are as follows:

Europe:	194.0.0.0 through to 195.255.255.255
North America:	198.0.0.0 through to 199.255.255.255
Central and South America:	200.0.0.0 through to 201.255.255.255
Asia and the Pacific:	202.0.0.0 through to 203.255.255.255

The remainder of the addresses are held in reserve.

As we can deduce from these allocations, the first byte in the destinat IP address indicates the continental backbone to which it should be se Hence the routing of packets across the global backbone network w addresses that are in this format is relatively straightforward. For examp any netid that contains 194 or 195 (in dotted decimal) as its first (dotte decimal number indicates the packet should be sent to the European ba bone. However, the absence of a fixed division point in the remaining 24 b means that each router in the backbone must be informed of the relat netid/hostid boundary before it can route the datagram any further.

The approach adopted is similar to that we described earlier in Secti 9.4.1 relating to subnetting. As we saw, with subnetting the hostid field is its divided into a subnetid part and a hostid part with no fixed bound between them. Instead, the division point is indicated by means of an addr mask which contains a binary 1 in all those bit positions that contain the su netid (and netid) fields. In a similar way, an address mask is used to indic the boundary between the netid and hostid parts of the new class addresses. Each exterior gateway then contains a copy of the address mask each of the networks within the autonomous systems that are attached t together with the base address – the netid – of the corresponding network

mple 9.4

A network attached to the Central and South Ameri [illegible] been allocated a block of 1024 contiguous class C add [illegible] 200.30.0.0 through to 200.30.3.255. Assuming the C [illegible] scheme, represent these addresses in binary form and [illegible] address mask to be used in dotted decimal form and [illegible]

Answer

Address 200.30.0.0 = 11001000 00011110 00000000 [illegible]

Address 200.30.3.255 = 11001000 00011110 0000001 [illegible]

Hence address mask = 11111111 11111111 111111100 [illegible]

= 255 255 252 [illegible]

and netid = 200.30.0.0

In this way, an exterior gateway, on receiving a packet, reads the destination IP address from the packet header and then performs the logical AND operation on this and the list of address masks that it contains. On detecting a match – that is, the resulting netid is the same as that stored with the corresponding mask – the exterior gateway uses the netid and the related interior gateway protocol to route the packet to the appropriate interior gateway. The packet is then passed on to the related access network gateway by this interior gateway.

As we can deduce from this, each interior gateway in the related autonomous system must also contain a copy of the address masks of the networks within that system. Also, each access gateway has a copy of its own address mask and, by using this, it first extracts the hostid from the destination IP address and then uses this to route the packet to that host.

Finally, as we can see from Example 9.4, it is possible for a number of hosts associated with a network which has been allocated a large block of addresses to produce a match with a mask relating to a network with a smaller block of addresses. However, since all network masks are tested, this will be in addition to the match relating to the mask with the smaller block of addresses. Should this happen, then the mask with the smaller block of addresses – and hence larger number of 1s in its address mask – is chosen as the most probable match.

Example 9.5

Two networks that are attached to the Central and South American backbone have been allocated the following block of class C addresses:

Network 1: Addresses = 200.64.16.0 through to 200.64.31.255
Mask = 255.192.16.0

Network 2: Addresses = 200.64.17.0 through to 200.64.17.255
Mask = 255.255.255.0

Assuming the CIDR addressing scheme, determine the address of a host attached to network 1 that will produce a match with the mask of network 2.

Answer

Network 1: Netid = 11001000 01000000 0001/xxxx xxxxxxxx

Network 2: Netid = 11001000 01000000 00010001/ xxxxxxxx

Hence

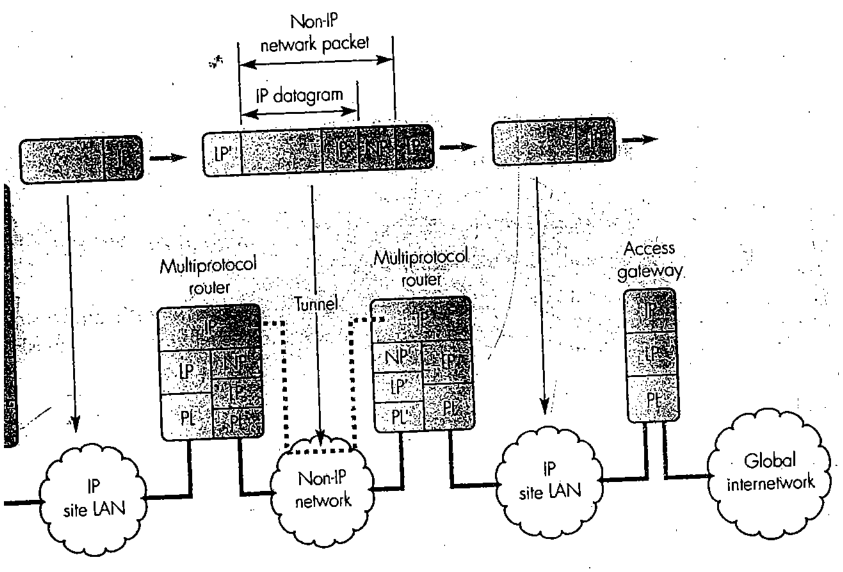
Network 1: Hostid = 11001000 01000000 0001/0001 xxxxxxxx

9.6.7 Tunneling

In the previous sections we have assumed that all networks within the global internetwork operate in a connectionless mode using the IP and its associated routing protocols. In practice, however, this is not always the case. As we showed earlier in Figure 9.9 and explained in the accompanying text, the Internet is made up of many separately managed networks and internetworks which, in some instances, use a different operational mode and/or protocol from the IP.

For example, consider a small enterprise consisting of two sites, both of which have LANs that operate using the TCP/IP stack, but only one of the sites has an access gateway connected to the Internet. Also, for cost reasons, instead of using a leased line to connect the two site LANs together, a public (or private) data network is used that operates in a connection-oriented mode and with a different protocol from the IP. Clearly, it is not possible to transfer each IP datagram directly over the public data network and instead a technique known as **tunneling** is used. Figure 9.20 illustrates this approach.

As we can see, in order to link the two sites together, a device known as a **multiprotocol router** is connected to each site LAN. As the name implies, a multi-protocol router operates using two different protocol stacks: the IP protocol stack on the site side and the protocol stack associated with the non-IP network on the other. The IP in each host simply treats the multiprotocol router as the site Internet access gateway. To send and receive packets to and from a host – a server for example – that is connected to the Internet, the IP simply sends the packet to the multiprotocol router using, for example, the ARP.



.20 Tunneling example.

Typically, the IP in the source router is given the (non-IP) network address of the multiprotocol router at the remote site by network management. On receipt of the packet, the IP in the source router, on determining that the netid in the destination IP address is not for this site, looks up the non-IP network address of the remote router and passes this, together with the datagram, to the network layer protocol associated with the non-IP network. The latter treats the datagram as user data and proceeds to transfer the datagram to the peer network layer in the remote router using the protocol stack of the non-IP network with the datagram encapsulated in a data packet relating to the network layer protocol.

On receipt of the data packet by the peer network layer protocol in the remote router, the user data – the datagram – contained within it is passed to the IP. The IP first determines from the destination IP address that the required host is not attached to the site LAN and hence proceeds to send the packet to the IP in the Internet access gateway using, for example, the ARP. A similar procedure is followed in the reverse direction to transfer the packets containing the related reply message. Thus, the presence of the non-IP network is transparent to the IP in each host and the access gateway.

In addition to using tunneling to transfer an IP packet over a non-IP network, the same technique is used to send an IP packet over an IP network. As we shall expand upon in Section 9.6.11, tunneling is used by an IP router to relay a packet that contains a multicast destination address to a different router that can handle multicast packets. Normally, the IP address of their nearest multicast router is known by all the other routers and, on receipt of a packet with a multicast address, the source router encapsulates the packet within a new packet with the IP address of the multicast router as the destination IP address.

9.6.8 Broadcast routing

As we explained in the last chapter, LANs such as Ethernet and token ring operate by a station/host broadcasting each frame over the LAN segment to which it is attached. The frame is then received by all the other stations that are attached to the same segment and, by examining the destination (MAC) address in the frame header, the network interface software within each host can decide whether to pass the frame contents on to the IP layer for further processing or to discard the frame. A frame is accepted if the destination MAC address is the same as its own individual address, or is a broadcast address, or is equal to one of the group addresses of which the station is a member. For a bridged LAN, this mode of working is extended to cover the total LAN by each bridge relaying all frames that contain either a broadcast or a multicast address on to all the other LAN segments to which the bridge is attached. In this section we explain how broadcasting is achieved at the I layer and, in the next section, how multicasting is achieved.

As we identified in Section 9.4, there are a number of different types o IP broadcast address:

- **limited broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the same LAN segment or bridged LAN. To achieve this, the destination IP address is set to 255.255.255.255.255. Neither subnet routers nor access gateways forward such packets;
- **subnet-directed broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the subnet specified in the destination IP address. To achieve this, the subnet mask associated with the subnet must be known and this is then used to determine the hostid part and set all these bits to 1. Such packets are forwarded by subnet routers but only if the destination netid is different from the source netid are they forwarded by access gateways, interior gateways, and, if necessary, exterior gateways;
- **net-directed broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the network specified in the netid part of the destination IP address. Such packets are forwarded by subnet routers but only if the destination netid is different from the source netid are they forwarded by access gateways, interior gateways, and, if necessary, exterior gateways.

Thus, a packet with a net-directed broadcast address whose destinatio netid is different from the source netid may need to be forwarded across th global internetwork. Since the destination netid is known, however, then th datagram can be routed by interior – and if necessary exterior – gateways i the same way as a packet with a unicast address. This also applies to a pack with a subnet-directed broadcast address whose destination netid is differe from the source netid. Also, since with a subnet-directed broadcast address a the subnet routers in both the source and destination networks have a co of the subnet mask, then they too can use the unicast routing algorithm as ciated with the network to route the packet to the subnet router specified the IP address. The latter then broadcasts the packet over this subnet. Wit net-directed broadcast, however, this is not possible and the unanswere question is how the packet is broadcast to all the subnets belonging to th network specified in the netid part of the address.

One solution is to use flooding but, as we concluded at the end of Sectio 9.6.2, this has very high bandwidth overheads associated with it. Two altern tive approaches are employed, the choice determined by the routin algorithm that is used to route unicast packets over the network. For descrip tion purposes we shall use the example of a large site/campus network th comprises a large number of subnets all interconnected by subnet routers

The aim of both algorithms is then for the arriving packet with a net-directed broadcast address to be broadcast over all the subnets using a minimum amount of bandwidth.

Reverse path forwarding

This algorithm is used primarily with networks that use the distance vector (DV) algorithm to route unicast packets. To explain the operation of the algorithm we use the network topology shown in Figure 9.21(a). We assume that subnet router 1 (SR1) also acts as the (single) access gateway for the network and that all subnets (SNs) are broadcast LANs.

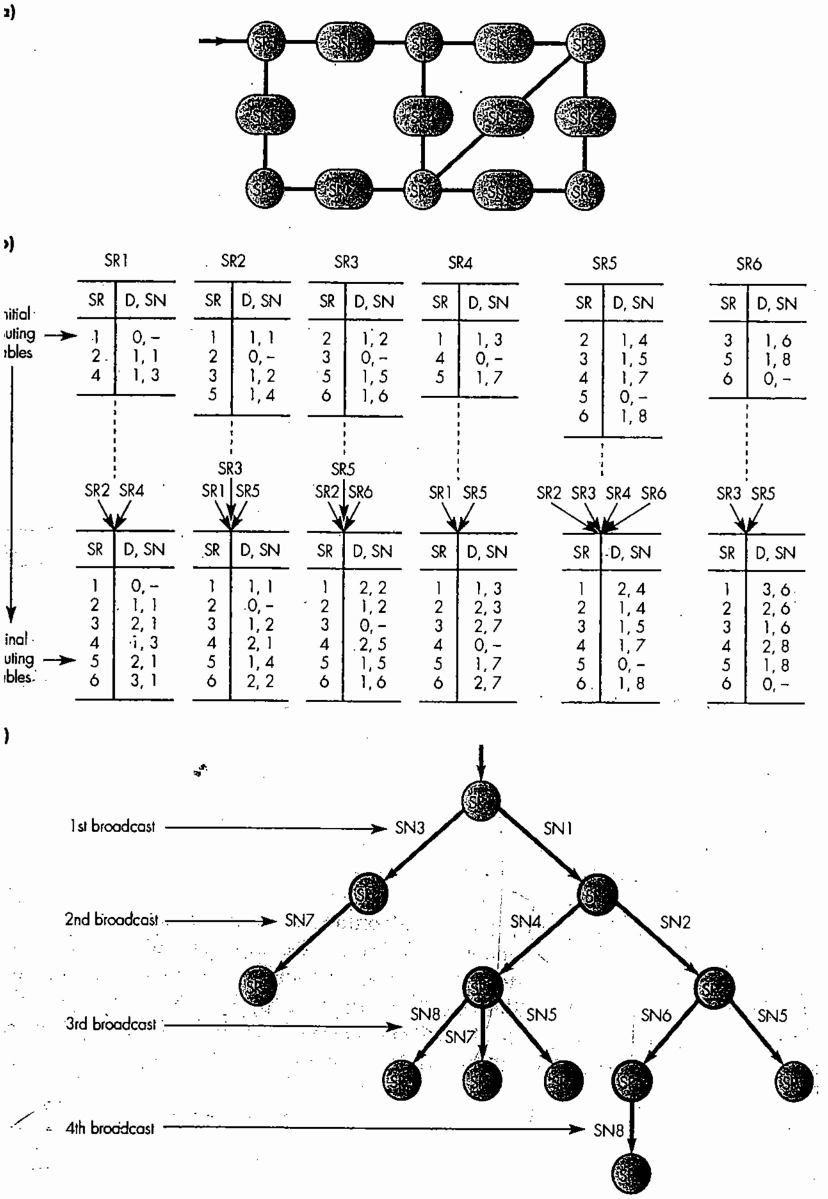
Using the DV algorithm we explained in Section 9.6.3, in addition to each subnet router deriving the shortest paths to each subnet, they can also derive the shortest paths to reach each of the other subnet routers. To see how this is done, the initial and final routing tables built up by each subnet router (based on a routing metric of hop count) are shown in Figure 9.21(b).

Once the routing tables have been created, the reverse path forwarding algorithm used to route (broadcast) packets works as follows. On receipt of a packet/datagram, the IP in each subnet router (SR) consults its routing table and only forwards a copy of it – onto each of the ports of the SR except the port the packet arrived on – if the packet arrived from an SR that is on the shortest path from SR1 to the SR that is processing the packet. If it is not, then the packet is discarded. Based on this simple rule, the path followed by each copy of an incoming packet is shown in Figure 9.21(c).

As we can see, on receipt of a packet, SR1 broadcasts a copy of it out onto subnets SN1 and SN3. Hence a copy of the packet is received by the IP in SR2 and SR4 respectively. On receiving the packet, the IP in SR2 first consults its routing table and determines from the (first) entry in the table that SN1 (from which the packet was received) is on the shortest path back to SR1. Similarly, when the IP in SR4 consults its routing table it also determines that SN3 (from which the packet was received) is also on the shortest path back to SR1. Hence both SR2 and SR4 are shown in bold in the figure and each proceeds to broadcast a copy of the packet, SR2 onto SN2 and SN4, and SR4 onto SN7.

After the second set of broadcasts, on receipt of its copy of the packet, the IP in SR3 determines from its routing table that SN2 is on the shortest path back to SR1. Similarly for the copy SR5 receives from SR2 via SN4. Hence both SR3 and SR5 are shown in bold and proceed to broadcast a copy of the packet, SR3 onto SN5 and SN6, and SR5 onto SN5, SN7, and SN8. However, in the case of the packet received by SR5 from SR4 via SN7, SR5 determines that SN7 is not on the shortest path back to SR1. Hence SR5 along this path is not shown in bold and the arriving packet is discarded.

The same procedure is repeated by SR5 and SR6 after the third set of broadcasts have been received but this time only SR6 determines from its routing table that SN6 is on the shortest path back to SR1 and proceeds to broadcast a copy of the packet onto SN8. The copies of the packet received



SR1

SR	D, SN
1	0, –
2	1, 1
4	1, 3

SR2

SR	D, SN
1	1, 1
2	0, –
3	1, 2
5	1, 4

SR3

SR	D, SN
2	1, 2
3	0, –
5	1, 5
6	1, 6

SR4

SR	D, SN
1	1, 3
4	0, –
5	1, 7

SR5

SR	D, SN
2	1, 4
3	1, 5
4	1, 7
5	0, –
6	1, 8

SR6

SR	D, SN
3	1, 6
5	1, 8
6	0, –

SR	D, SN
1	0, –
2	1, 1
3	2, 1
4	1, 3
5	2, 1
6	3, 1

SR	D, SN
1	1, 1
2	0, –
3	1, 2
4	2, 1
5	1, 4
6	2, 2

SR	D, SN
1	2, 2
2	1, 2
3	0, –
4	2, 5
5	1, 5
6	1, 6

SR	D, SN
1	1, 3
2	2, 3
3	2, 7
4	0, –
5	1, 7
6	2, 7

SR	D, SN
1	2, 4
2	1, 4
3	1, 5
4	1, 7
5	0, –
6	1, 8

SR	D, SN
1	3, 6
2	2, 6
3	1, 6
4	2, 8
5	1, 8
6	0, –

21 Reverse path forwarding: (a) network topology; (b) distance vector routing ing a hop-count metric; (c) broadcast sequence.

from SR5 by SR3, SR4, and SR6 are all discarded as the related subnets – SN5, SN7, and SN8 – are not on the shortest paths back to SR1. Also the packet received by SR5 from SR6 after the fourth broadcast.

As we can deduce from this example, a copy of the packet is broadcast over all the eight subnets that make up the network and only SN7 and SN8 receive two copies. However, since both copies of the packet have the same value in the identifier field of the packet header, the second copy can be detected by each of the hosts on these subnets as a duplicate and is discarded. Note also that the same set of routing tables can be used to perform the same algorithm if a second (or different) SR acts as an additional (or alternative) access gateway. Also, if the broadcast is over the source network.

Spanning tree broadcast

With networks that use a routing algorithm based on the link-state algorithm, an alternative way of routing broadcast packets/datagrams is for each router to use the link-state information to establish a spanning tree active topology with the access gateway/router as the root node.

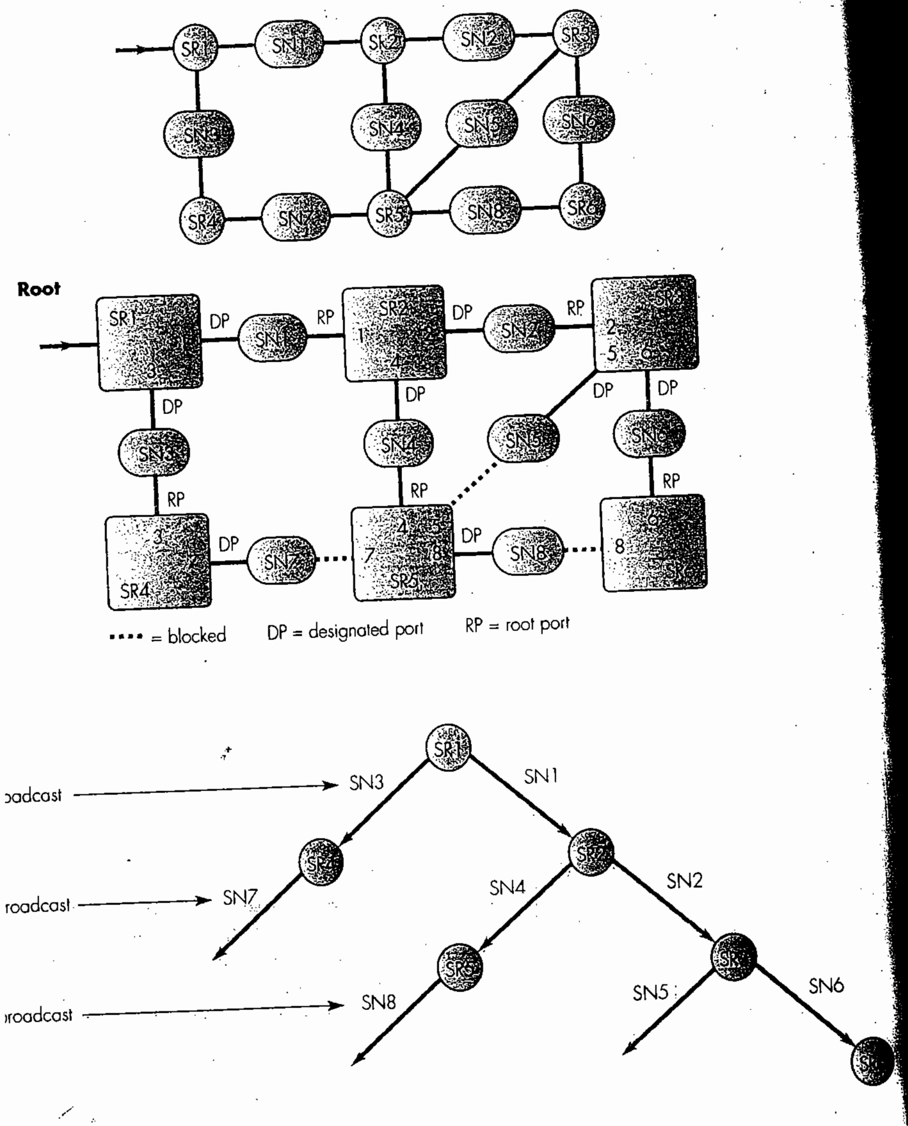
As we saw in Section 9.6.4, the information gathered as part of the link-state algorithm is used by each router to derive the current active topology of the internetwork. In a similar way, therefore, with networks that consist of multiple subnets interconnected by subnet routers, each subnet router builds up knowledge of the current active topology of the network and then uses this to compute the shortest path to reach all the subnetids in the network.

Hence with the spanning tree broadcast algorithm, in addition to each subnet router computing the shortest paths, they all derive the (same) spanning tree topology from the current active topology using, for example, the algorithm we described in Section 8.5.1. As we saw, a spanning tree topology is established in order to avoid frames looping within a bridged LAN. This is done by defining the ports associated with each bridge as either root ports or designated ports. All ports that are either root or designated ports are then set into the forwarding state and all the other ports are set into the non-forwarding (blocked) state.

Using the same approach, we can derive a spanning tree by setting selected subnet router ports into the forwarding and blocked state. For example, using the network topology we showed in Figure 9.21(a) and the algorithm we described in Section 8.5.1, assuming each subnet router knows the root SR, each will derive the spanning tree shown in Figure 9.22(b). The resulting broadcast sequence is therefore as shown in Figure 9.22(c).

We can make a number of observations from this example:

- For consistency, the port numbers associated with each subnet router are determined by the (known) identifier of the attached subnet.
- Each SR has a root port (RP) associated with it which is the port with the shortest path back to the root. The path costs in the example are based on hop count and, in the event of a tie, the port with the smallest port number is chosen.



.22 Spanning tree broadcast: (a) network topology; (b) spanning tree derived by net router; (c) broadcast sequence.

- For each subnet, there is a designated port (DP) which is the router port on the shortest path from the root to the subnet. In the event of a tie, the SR with the smallest identifier is chosen.
- All router ports that are not root or designated ports (DP) are set into the blocked state.
- Only a single copy of each received packet/datagram is broadcast over each subnet.
- The same spanning tree can be used to broadcast packets if a second (or different) SR acts as an additional (or alternative) access gateway and if the broadcast is over the source network.

9.6.9 Multicast routing

As we explained in Chapter 1, applications such as audio- and videoconferencing require a copy of the information generated by each host participating in a conference to be sent to all the other hosts that belong to the same conference. The term **multicasting** is used to describe the diffusion of a copy of the packets/datagrams generated by each host to all the other hosts and the term **multicast group** is used to identify the hosts that are members of the same conference. Clearly, there can be many conferences in progress concurrently, each involving a different group of hosts. It is to support applications of this type that class D multicast addresses were defined. As we showed in Figure 9.4 and explained in the accompanying text, a single class D address is used to identify all the hosts that are members of the same multicast group. Also, since 28 bits are available, many different multicast groups can be in place concurrently.

A number of multicast addresses are reserved to identify specific groups. For example, all the hosts (and subnet routers) that are attached to a site broadcast network are members of the group with the reserved multicast address of 224.0.0.1. Hence the IP in all hosts that can support multicasting will have this address permanently in their **multicast address table (MAT)** and, should a packet be received with this address, the datagram contents will be passed to (and acted upon) by a related application process. Similarly, all the subnet routers belonging to the same network are members of the multicast address 224.0.0.2.

As we showed in Figure 5.4, with networks that do not support broadcasting such as PSTNs and ISDNs, the diffusion operation is achieved by using a central (conference) server known as a multipoint control unit (MCU). Also, as we showed in Figure 5.6, in packet-switching networks that support broadcasting such as LANs, the H.323 standard uses a similar approach with a networked device known as a gatekeeper performing the multipoint control function of an MCU. Using this approach, the IP and MAC address-pair of the MCU is known by the IP in all the hosts that are participating in a conference and, once the IP in the MCU has obtained the IP/MAC address-pair of all the participating hosts, the IP in the MCU can carry out the diffusion operation using only the individual (unicast) IP address of each participant.

This approach is acceptable providing the number of participating hos is relatively small and the composition of the conference is static. Howeve for applications such as a conference or meeting which is being transmitte over a LAN or the Internet, a relatively large number of participants – an hence hosts – may be involved. Moreover, hosts may wish to join and/or leav the conference/meeting whilst it is in progress. It is to meet this type of appl cation that multicasting using IP multicast addresses is used.

With this mode of working, the organizer of the conference/meetin first obtains an IP multicast address for it from the Internet assigned numbe authority (IANA). The allocated address is then made known to all the regi tered participants together with the conference/meeting start time and i likely duration. Each participating host can then request to join the confe ence at any time during the time the conference/meeting is in progress. T do this, two operating scenarios are used which depend on whether the pa ticipating hosts are all attached to the same LAN/subnet or, as is more usua are attached to many different networks that are geographically distribute around the Internet. We shall discuss each separately.

Multicasting over a LAN

The IANA has been allocated a block of Ethernet MAC addresses for applica tions that involve multicasting. As we described in Section 8.3, Ethernet MAC addresses are 48 bits in length and the reserved block of addresses in dotte decimal are from

0.0.94.0.0.0 through to 0.0.94.255.255.255

Half of these addresses are used for multicasting. Hence, rememberin that all centrally administered Ethernet group (multicast) addresses mus start with 10 binary, the block of Ethernet addresses used for centrally admin istered multicasting applications are from

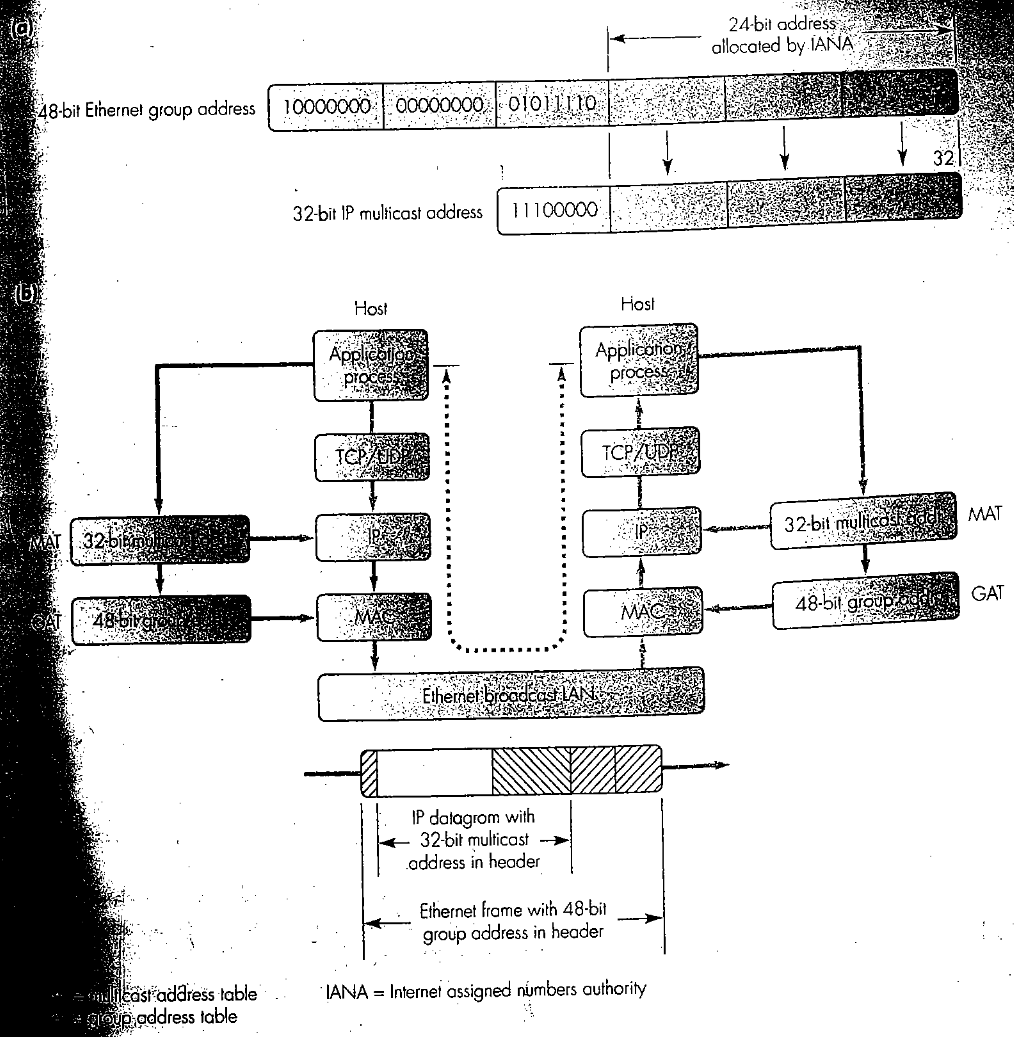
128.0.94.0.0.0 through to 128.0.94.127.255.255

Thus for a particular conference/meeting, a 3-byte address in the range

0.0.0 through to 127.255.255

is allocated by the IANA. In the case of an Ethernet LAN, this forms the lea significant 24 bits of the 48-bit destination MAC address – starting with th three bytes 128.0.94 – and, in the case of the Internet, the least significan 24 bits of the destination IP multicast address. As we explained earlier, clas D IP addresses are 28 bits in length – the first four bits being 1110 – an hence, as we show in Figure 9.23(a), the four remaining bits are all set to 0.

So to join a conference/meeting that is being broadcast over the same LAN, once the multicast address is known, the application process running in


(a)
24-bit address allocated by IANA
48-bit Ethernet group address
10000000
00000000
01011110
32
32-bit IP multicast address
11100000
(b)
Host
Host
Application process
TCP/UDP
IP
MAC
MAT
32-bit multicast address
GAT
48-bit group address
Ethernet broadcast LAN
IP datagram with 32-bit multicast address in header
Ethernet frame with 48-bit group address in header
multicast address table
group address table
IANA = Internet assigned numbers authority

9.23 Multicasting over a LAN: (a) address allocation principle; (b) address usage.

the host that is managing the information associated with the conference/meeting simply loads the allocated IP multicast address into its MAT and the related 48-bit Ethernet group address derived from this into its **group address table** (GAT). Each datagram relating to the conference/meeting then has the allocated multicast address in the destination IP address field of the datagram header. Each packet is then broadcast over the LAN i an Ethernet frame containing the derived 48-bit group address in the destination MAC address field of the frame header.

On receipt of a frame, the MAC sublayer in each host that is a member o the same multicast group, first checks to see whether the destination MA address in the frame header is present in its group address table and, if it is, i passes the frame contents – the datagram – on to the IP layer. The latter firs determines that the destination IP address is a multicast address and also tha it is present in its multicast address table. It therefore passes the informatio contained within the datagram on to the related application process via the TCP or UDP. This procedure is shown in Figure 9.23(b).

Multicasting over the Internet

When the hosts that are part of a multicast group are attached to different networks/subnetworks geographically distributed around the Internet, then intermediate subnet routers and/or interior/exterior gateways may be involved. Thus, since an IP multicast address has no structure – and hence no netid – associated with it, a different type of routing from that used to route unicast packets must be used.

The sequence of steps followed to route a packet with a multicast address is as follows:

- A router that can route packets containing a (destination) IP multicast address is known as a **multicast router (mrouter)**.
- Normally, in the case of a network that comprises multiple subnets interconnected by subnet routers, a single subnet router also acts as the mrouter for that network.
- Each mrouter learns the set of multicast group addresses of which all the hosts attached to the networks which the mrouter serves are currently members.
- The information gathered by each mrouter is passed on to each of the other mrouters so that each knows the complete list of group addresses that each mrouter has an interest in.
- On receipt of a packet with a destination IP multicast address, each mrouter uses an appropriate routing algorithm to pass the packet only to those mrouters that are attached to a network which has an attached host that is a member of the multicast group indicated in the destination IP address field.

As with broadcast routing, two different algorithms are used, the choice determined by the routing algorithm that is used to route unicast packets. The aim of both algorithms is to minimize the amount of transmission bandwidth required to deliver each multicast packet to those multicast routers that

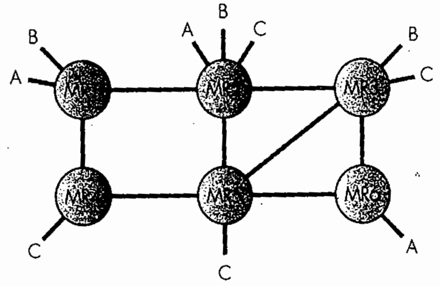
---

have an interest in the packet. To explain the two algorithms, we shall use the internetwork topology shown in Figure 9.24(a). The letters by each multicast router (MR) – A, B, and C – indicate the multicast address(es) that each has an interest in and, since all MRs exchange this information, they each have a copy of the multicast address table shown in Figure 9.24(a).

DVMRP

When distance vector routing is being used, an additional set of routing tables (to those used to route unicast packets) based on MR-to-MR distances are derived and, for the six MRs, these are given in Figure 9.24(b). They are based on a routing metric of hop count and have been derived using the same procedure we described earlier in Section 9.6.3. Together with the contents of the MAT in each MR, they are used to route all packets that have a destination multicast address.

(a)



Multicast address table

MR	Multicast address
1	A, B
2	A, B, C
3	B, C
4	C
5	C
6	A

(b)

MR2 ↘ MR1 ↙ MR4

MR	MR, D
1	1, 0
2	2, 1
3	2, 2
4	4, 1
5	2, 2
6	2, 3

MR1 MR3 MR5 → MR2

MR	MR, D
1	1, 1
2	2, 0
3	3, 1
4	1, 2
5	5, 1
6	3, 2

MR2 MR5 MR6 → MR3

MR	MR, D
1	2, 2
2	2, 1
3	3, 0
4	5, 2
5	5, 1
6	6, 1

MR1 ↘ MR4 ↙ MR5

MR	MR, D
1	1, 1
2	5, 2
3	5, 2
4	4, 0
5	5, 1
6	5, 2

MR2 MR4 MR6 → MR5

MR	MR, D
1	2, 2
2	2, 1
3	3, 1
4	4, 1
5	5, 0
6	6, 1

MR3 ↘ MR6 ↙ MR5

MR	MR, D
1	3, 3
2	3, 2
3	3, 1
4	5, 2
5	5, 1
6	6, 0

MR = multicast router D = distance in hops

Figure 9.24 Distance vector multicast routing: (a) example topology and multicast address table; (b) unicast routing tables of MR1–6.

The protocol is known as the **distance vector multicast routing protocol** (**DVMRP**). To explain how it works, assume a packet with a multicast address of A is received by the IP in MR1 from one of its attached networks. The sequence is as follows:

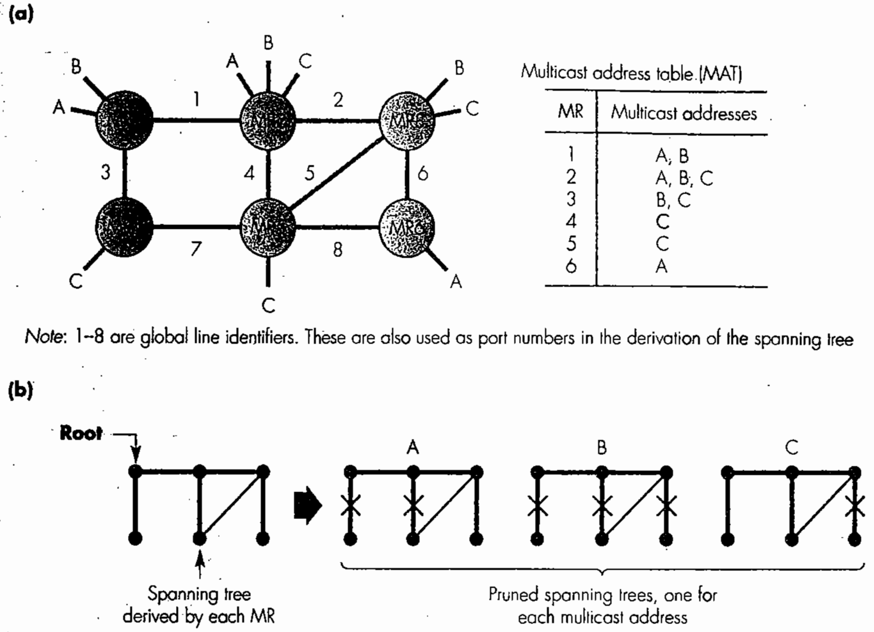
- The IP first consults its MAT and finds that a copy of the packet should be sent to MR2 and MR6. It then proceeds to consult its routing table (RT) and finds that the shortest path to MR6 is also via MR2 and hence sends just a single copy of the packet to MR2.
- On receipt of the packet, MR2 consults its MAT and sees that a copy of the packet should be sent out onto its own network and also to MR1 and MR6. On consulting its RT, however, it finds that the shortest path to MR1 is on the line/port the packet was received and hence it only sends a copy of the packet to MR6. The RT indicates that the shortest path to MR6 is via MR3 and hence it forwards a copy of the packet to MR3.
- On receipt of the packet, MR3 determines from its MAT that MR1, MR2, and MR6 should be sent a copy of the packet. On consulting its RT it finds that the shortest path to both MR1 and MR2 is via MR2. Hence, since this is the line/port the packet arrived on, it only sends a copy of the packet to MR6.
- On receipt of the packet, MR6 determines from its MAT that a copy of the packet should be sent out on its own network and also to MR1 and MR2. However, since the shortest path to both MR1 and MR2 is via the line/port the packet arrived on, it only sends a copy out on its local network.

MOSPF

When link-state routing is being used, since each MR knows the current active topology of the internetwork, as we explained in Section 9.6.8, each computes a spanning tree for the internetwork using the algorithm we described in Section 8.5.1. For example, for the internetwork shown in Figure 9.25(a), the spanning tree computed by each MR is shown in Figure 9.25(b). The numbers shown by each line are used as global line identifiers and as port numbers in the derivation of the spanning tree.

At any point in time, each MR knows from its MAT the multicast addresses that each has an interest in. For each address – A, B, and C in the example – each MR proceeds to produce a pruned spanning tree by removing selected links. For the internetwork shown in Figure 9.25(a), the pruned spanning trees produced by each MR for each of the three multicast addresses are as shown in Figure 9.25(b).

For each address, these are produced by starting at the tip of each branch of the basic spanning tree and pruning each line that does not form a path back to the root for this address. Packets are then only forwarded on those lines that make up the resulting pruned spanning tree. The protocol is known as the **multicast open shortest path first (MOSPF)** routing protocol


(a)
A
B
C
1
2
3
4
5
6
7
8
MR3
MR6
(b)
Root
A
B
C
Spanning tree derived by each MR
Pruned spanning trees, one for each multicast address

Multicast address table (MAT)

MR	Multicast addresses
1	A, B
2	A, B, C
3	B, C
4	C
5	C
6	A

Note: 1–8 are global line identifiers. These are also used as port numbers in the derivation of the spanning tree

Figure 9.25 Spanning-tree multicast routing: (a) example topology and multicast address table; (b) set of spanning trees for each multicast address.

and, to explain how it works, assume that a packet with a (multicast) address of A is received from a network attached to MR1. The steps followed by each MR are as follows:

- On detecting the (multicast) address in the packet header is A, MR1 uses the pruned spanning tree for A to determine that a copy of the packet should be sent to MR2.
- On receipt of the packet, MR2 first determines from its MAT that a copy of the packet should be sent out onto its own network. It then uses its own copy of the pruned spanning tree for A to determine that a copy of the packet should also be sent to MR3.
- On receipt of the packet, MR3 first determines from its MAT that it has no interest in the packet. It then uses its own copy of the pruned spanning tree for A to determine that a copy of the packet should be sent to MR6.
- On receipt of the packet, MR6 first determines from its MAT that a copy of the packet should be sent out onto its own network. It then uses its own copy of the pruned spanning tree for A to determine that it is at the end of a branch of the tree and hence discards the packet.

9.6.10 IGMP

The two routing algorithms we described in the last section – the **DVMRP** a MOSPF – both assumed that each multicast router has stored in its multica address table the complete list of multicast addresses that are currently in u and also the multicast addresses that each multicast router has an interest i As we explained, the contents of the MAT are obtained by, first, each multicas router learning the set of multicast addresses of which all the hosts attached t the networks/subnets which the mrouter serves are currently members an second, this information is passed on to all the other mrouters. Clearly, th second step can be carried out using a broadcast procedure similar to that used in the distance vector algorithm to distribute network-wide routing information. Hence the unanswered question is how an mrouter learns the multicast addresses associated with its own attached networks/subnets. In practice, this is the role of the Internet group management protocol (IGMP) which we identified earlier in Figure 9.2. The IGMP is an integral part of the IP layer in all hosts and routers that support multicasting. It is defined in **RFC 1112.**

As we explained at the start of the last section, an application process (AP) within a host can join and leave a currently active multicast group at any time. To do this, the AP must know the 24-bit group address that has been allocated to this group by the IANA. As we showed in Figure 9.23, the AP uses this to derive both the IP multicast address of the group and the corresponding Ethernet group address. It then writes these into the MAT and the corresponding GAT respectively. In the case of a multicast session involving a group of APs in hosts that are all attached to the same LAN, this is all that is needed. In the case of a multicast session over the Internet, however, it is necessary for the host to inform its local mrouter the multicast address of the session that it wishes to join. The sequence of steps followed to do this are shown in Figure 9.26.

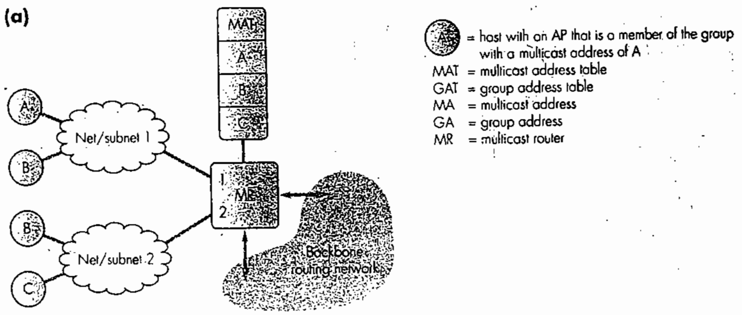

(a)
MAT
A
B
C
A
B
B
C
Net/subnet 1
Net/subnet 2
1
MR
2
Backbone routing network
= host with an AP that is a member of the group with a multicast address of A
MAT = multicast address table
GAT = group address table
MA = multicast address
GA = group address
MR = multicast router

Figure 9.26 IGMP summary: (a) example network topology; (b) IGMP message transfer sequence to join and leave a multicast session; (c) IGMP message format.

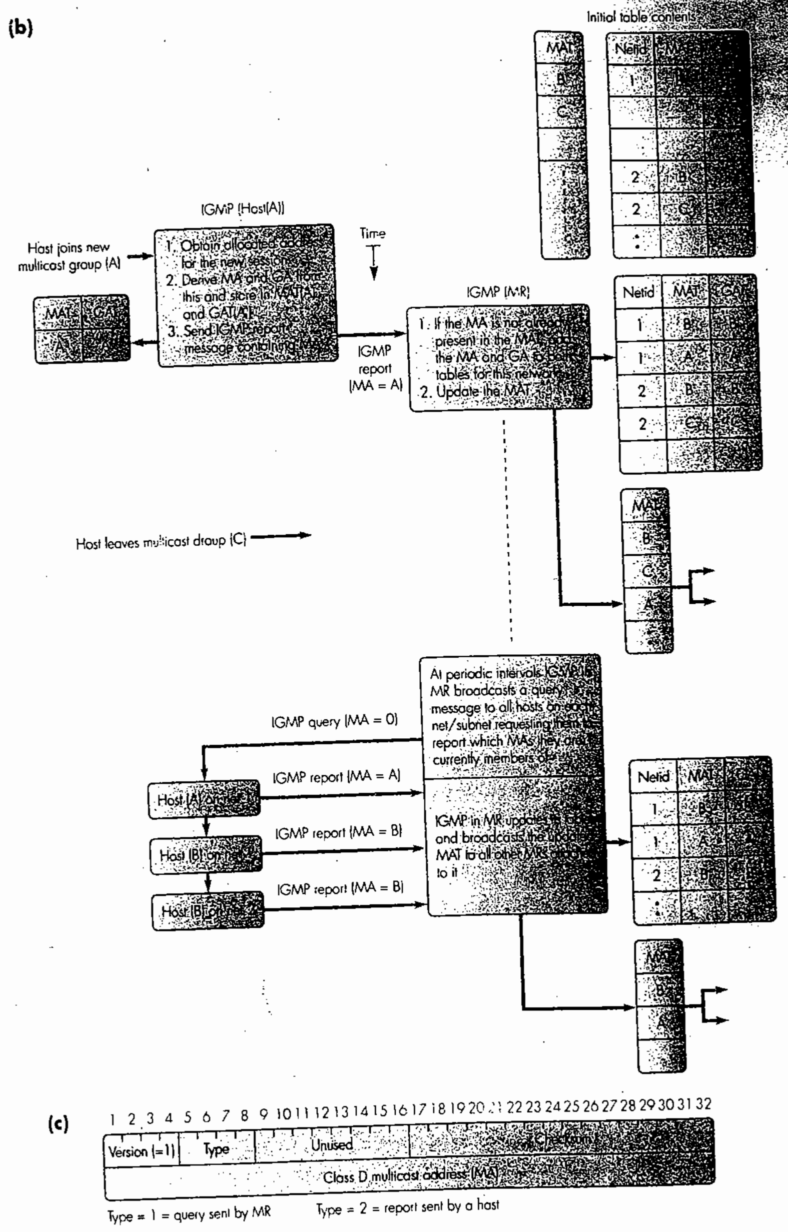

(b)
Initial table contents
MAT
B
C
Netid
MA
1
B
2
B
2
C
:
Host joins new multicast group (A)
IGMP (Host(A))
1. Obtain allocated add for the new session
2. Derive MA and GA from this and store in MAT(A) and GAT(A)
3. Send IGMP report message containing MA
MAT
GAT
A
A
Time
IGMP report (MA = A)
IGMP (MR)
1. If the MA is not already present in the MAT, add the MA and GA to both tables for this network
2. Update the MAT
Netid
MAT
GAT
1
B
1
A
2
B
2
C
MAT
B
C
A
Host leaves multicast droup (C)
At periodic intervals IGMP in MR broadcasts a query message to all hosts on each net/subnet requesting them to report which MAs they are currently members of
IGMP query (MA = 0)
IGMP report (MA = A)
IGMP report (MA = B)
IGMP report (MA = B)
Host (A) on net 1
Host (B) on net 1
Host (B) on net 2
IGMP in MR updates its tables and broadcasts the updated MAT to all other MRs attached to it
Netid
MAT
1
B
1
A
2
B
:
MAT
B
A
(c)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Version (=1)
Type
Unused
Checksum
Class D multicast address (MA)
Type = 1 = query sent by MR
Type = 2 = report sent by a host

Figure 9.26 Continued

In this example, an AP running in a host that is attached to net/subne wishes to take part in a multicast session that has a derived multicast addr of A. We assume that three other APs/hosts attached to the same multic router are already members of two existing multicast groups, two with a mul cast address of B and one with an address of C. In the case of B, one attached to net/subnet 1 and the other to net/subnet 2 and, in the case of this is attached to net/subnet 2. Hence the contents of the initial tables are shown in Figure 9.26(b).

First the IGMP in host (A) sends out a message – known as a *report* – the IGMP in the attached mrouter (MR) containing the multicast addre (MA) of the group it wishes to join (A). In addition to the MAT that is use for routing over the backbone network, a separate MAT and GAT are mai tained by the MR for both net/subnet 1 and net/subnet 2. Hence on recei of the report message, the IGMP first writes the MA into the MAT and th related Ethernet group address into the GAT of net/subnet 1. Also, since A not already present in the backbone routing MAT, this is added to it and th new contents forwarded to each of the attached MRs.

Hosts do not need to inform an MR when it leaves a multicast group. A regular intervals, the IGMP in each MR broadcasts a *query* message to all hos on each net/subnet requesting them to report of which MAs they are cu rently members. On receipt of a query, the IGMP in each host responds b returning a separate report message for each MA of which the host is cu rently a member. In the example shown in Figure 9.26, we assume the hos that was a member of MA C has now left the group and hence a report me sage for C is not returned. Hence the IGMP in the MR removes the entri for C from its tables and proceeds to forward the updated table contents t each of the attached MRs.

The format of the two types of IGMP message – query and report – is th same and is shown in Figure 9.26(c). The *version* field is equal to 1 and the *ty* field is either 0 for a report message (sent by a host) or 1 for a query messag (sent by a MR). The *checksum* applies to the complete message and is compute using the same 1s complement procedure used with the header of an IP da gram. The *group address* is a 32-bit class D multicast address. In a query messag it is set to all 0s and in a report it is set to the multicast group address bein reported. Both are transmitted over the attached net/subnet in an IP pack with a *protocol* value of 2 and a *time-to-live* value of 1. In the case of a query, th *destination IP address* is the all-hosts broadcast address of 224.0.0.1 and the *sou* *IP address* is the IP address of the MR. In the case of a report, the two addresse are the group MA being reported and the host IP address respectively.

9.6.11 M-bone

As we explained in Section 9.6.9 when we discussed the two multicast routin algorithms, currently, only a subset of the routers that make up the Interne global internetwork are capable of routing IP packets which have a multicas

destination address. These are known as multicast routers (mrouters) and the network formed by the interconnected set of mrouters, the **multicast backbone (M-bone) network**. The two routing algorithms we described in Section 9.6.9 are then used to route multicast packets between the mrouters that make up the M-bone.

In practice, therefore, because only a subset of the routers in the global internetwork can route multicast packets, there may be other routers that do not support multicasting present in the physical path that links two mrouters together. Hence many of the transmission paths that link the mrouters that form the M-bone are logical links that are implemented using IP tunneling. As we explained in Section 9.6.7, with tunneling, in order to send an IP datagram containing a multicast group address over the logical link that links two mrouters together, the datagram is carried in the user data field of a second IP datagram. This has the (known) IP unicast address of the intended destination mrouter in the destination IP address field of the second datagram. In this way, the packet is routed over the global internetwork in the same way as a unicast packet using either the distance vector or the link-state shortest-path-first algorithm. Then, on arrival at the destination mrouter, the latter extracts the multicast datagram contained within the packet and proceeds to route it using one of the two related multicast routing algorithms we described in Section 9.6.9.

As we can deduce from this, in order to implement the M-Bone, each mrouter has the IP (unicast) address of the adjacent mrouters that are (logically) connected to it stored in their connectivity/adjacency table. Normally, this information is entered by the management authority responsible for the network in which the mrouter is located. In this way, the possible presence of other (unicast) routers between two mrouters is transparent to the two mrouters which simply route (multicast) packets as if there is a physical transmission line linking them together.

9.7 ICMP

The Internet control message protocol (ICMP) forms an integral part of all IP implementations. It is used by hosts, routers and gateways for a variety of functions, and especially by network management. The main functions associated with the ICMP are as follows:

- error reporting,
- reachability testing,
- congestion control,
- route-change notification,
- performance measuring,
- subnet addressing.

The message types associated with each of these functions are shown in Tab
9.2. Each is transmitted in a standard IP datagram.

Since the IP is a best-effort (unacknowledged) protocol, packets may
discarded while they are in transit across the Internet. Of course, transmissio
errors are one cause, but packets can be discarded by a host, router, or gatew
for a variety of reasons. In the absence of any error reporting functions, a ho
does not know whether the repeated failure to send a packet to a given destin
tion is the result of a poor transmission line (or other fault within a network) o
simply the destination host being switched off. The various messages associate
with the **error reporting** function are used for this purpose.

Table 9.2 ICMP message types and their use

Function	ICMP message(s)	Use
Error reporting	Destination unreachable	A datagram has been discarded due to the reaso specified in the message
	Time exceeded	Time-to-live parameter in a datagram expired and hence discarded
	Parameter error	A parameter in the header o datagram is unrecognizable
Reachability testing	Echo request/reply	Checks the reachability of a specified host or gateway
Congestion control	Source quench	Requests a host to reduce the rate at which datagrams are sent
Route exchange	Redirect	Used by a gateway to inform host attached to one of its networks to use an alternative gateway on the same network for forwardin datagrams to a specific destination
Performance measuring	Time-stamp request/reply	Determines the transit delay between two hosts
Subnet addressing	Address mask request/reply	Used by a host to determine the address mask associated wit a subnet

ICMP = Internet control message protocol

If a packet is corrupted by transmission errors, it is simply discarded. If a packet is discarded for any other reason, the ICMP in the host, router, or gateway that discards the packet generates a *destination unreachable* error report message and returns it to the ICMP in the source host with a reason code. Reasons include the following:

- destination network unreachable,
- destination host unreachable,
- parameter problem,
- specified protocol not present at destination,
- fragmentation needed but don't fragment (DF) flag set in datagram header,
- communication with the destination network not allowed for administrative reasons,
- communication with the destination host not allowed for administrative reasons.

Although in most cases error reports are received as a result of some type of failure condition arising, in some instances an error report is used to gain some knowledge of the operational characteristics of the path/route followed by a packet. In general, the transfer of a message over the global internetwork is much quicker if no fragmentation is involved. Most networks (and subnetworks) support a maximum transmission unit (MTU) equal to or greater than 576 bytes. Hence one way of ensuring no fragmentation takes place is for the source IP to adopt this as the maximum size of all datagrams (including the IP header). In most cases, however, the actual MTU of the path will be greater than this and hence this would result in more packets being used to send each message than is necessary.

An alternative is for the source IP to use a procedure known as **path MTU discovery** to determine the MTU of a path/route prior to sending any datagrams relating to a session/call. Essentially, the first message received from the transport layer protocol relating to a new call/session is sent in a single datagram with the *don't fragment* bit set. Normally, if a router along the path followed by the packet cannot forward the packet over an attached link without fragmenting it, the router will return an ICMP error report with *fragmentation needed* as a reason code and an indication of the size of MTU that is possible. The source IP then adopts the latter as its own MTU to send all the remaining messages relating to the call/session.

Other error report messages include *time exceeded*, which indicates that the time-to-live parameter in a discarded packet has expired, and *parameter error*, which indicates that a parameter in the header of the discarded packet was not recognized.

If a network manager receives reports from a user that a specified destination is not responding, the reason must be determined using the *reachability testing* function which is implemented in a program called ping. Typically, on receipt of such a report, the network manager initiates the sending of an *echo request* message to the suspect host to determine whether it is switched on and responding to requests. On receipt of an echo request message, the ICMP in the destination simply changes this to an *echo reply* message which it returns to the source. A similar test can be performed on selected routers and gateways if necessary.

If a packet is discarded because no free memory buffers are available as a result of a temporary overload condition, a *source quench* message is returned to the ICMP in the source host. Such messages can be generated either by a host or by a router or gateway. They request the source host to reduce the rate at which it sends packets. When a host receives such a message, it reduces the sending rate by an agreed amount. A new source quench message is generated each time a packet is discarded so that the source host incrementally reduces the sending rate. Such messages help to alleviate congestion within the global internetwork. Congestion is discussed further in Section 9.8.

When a network has multiple gateways attached to it, a gateway may receive datagrams from a host even though it determines from its routing table that they would be better sent via a different gateway attached to the same network. To inform the source host of this, the ICMP in the gateway returns a *redirect* message to the ICMP in the source indicating which is the better gateway to the specified destination. The ICMP in the source then makes an entry in its routing table for this destination.

An important operational parameter for an internet is the mean transit delay of packets/datagrams. This is a measure of the time a datagram takes to traverse the internet from a specified source to a specified destination. To ascertain this time, a host or a network manager can send a *time-stamp request* message to a specific destination. Each message contains the following three time-related parameters (known as *time-stamps*):

- the time the datagram was sent by the source,
- the time the datagram was received by the destination,
- the time the datagram was returned by the destination.

On receipt of a time-stamp request message, the ICMP in the destination simply fills in the appropriate time-stamp fields and returns the message to the source. On receipt of the reply, the source can quantify the current round-trip delay to that destination and from this determine the packet transit delay.

Finally, when subnet addressing is being used, the *address mask request* and corresponding reply messages are used by a host to ascertain the address mask associated with a local subnet. This is needed by a host to determine, for example, whether a specified destination is attached to the same subnet. The address mask is held by the local router associated with the subnet. The ICMP in a host can obtain the address mask by sending a request message and reading the mask from the reply.

9.7.1 ICMP message formats and transmission

The format of an ICMP message is shown in Figure 9.27. The first three fields are the same for all messages. The *type* field indicates the ICMP message type and these are related to the functions we listed earlier. For example, a *type* field of 0 relates to the error reporting function, 3 reachability testing, 4 congestion control, and so on. The *code* field then gives additional information such as the reason why a destination is unreachable. For example, 6 indicates the destination network is unknown and 7 the destination host is unknown. The *checksum* covers the entire ICMP message and uses the same algorithm as that used for the checksum present in the IP header. The number and meaning of the following 32-bit words then depend on the *type* and *code* fields in the header.

As we showed in Figure 9.3 and explained in the accompanying text, the *protocol* field in each datagram header is used to route the payload in a datagram to the appropriate protocol – ICMP, IGMP, OSPF, TCP, or UDP. In the case of reply messages, normally, these are for use by the local IP. When unsolicited error reports are received, in most cases, they are passed to an application-level process and result in an appropriate error message being output on the host screen.

9.8 QoS support

Congestion arises within a network when the demand for a network resource exceeds the level that is provided. For example, if a burst of packets arrive at a router (within the global internetwork) on a number of different input lines that all require the same output line, then the output line will become

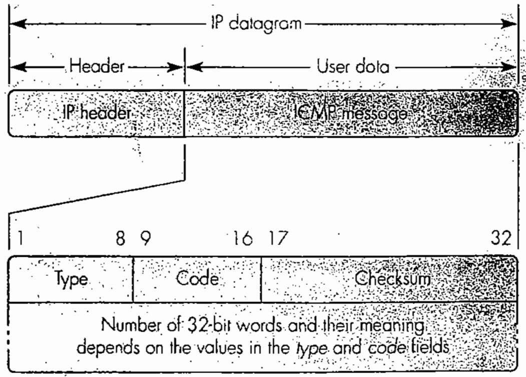

IP datagram
Header
User data
IP header
ICMP message
1
8 9
16 17
32
Type
Code
Checksum
Number of 32-bit words and their meaning depends on the values in the type and code fields

Figure 9.27 ICMP message format and transmission.

congested if the rate of arrival of packets is greater than the rate they ca
output. To allow for this possibility, each output line has a first-in first-
(FIFO) queue associated with it which is used to hold a defined numbe
packets that are awaiting output on that line. Hence, providing the burst i
a relatively short duration and the number of packets to be queued is l
than the number of packet buffers available, the congestion will be transi
and the only effect should be a small increase in the end-to-end transfer de
experienced by each packet using that line. In the event of a longer bu
however, then all the packet buffers may become full and, as a result, so
packets will have to be discarded.

Similarly, at a network-wide level, since the Internet is a best-effort co
nectionless network, the global internetwork will become congested if, ov
a sustained period, the aggregate rate at which packets are entering t
internetwork exceeds its total capacity in terms of transmission bandwid
and packet buffers. As we saw in Section 1.5.5, associated with each call i
defined set of parameters which form what is called the minimum quality
service (QoS) requirements for the call. For example, with a packe
switched network like the Internet, these include a defined minimum me
packet throughput rate and a maximum end-to-end packet transfer dela
Hence, if as a result of congestion these requirements are not met, then t
quality of the call may no longer be acceptable to the user. This is the ca
with applications involving real-time media streams, for example, such
Internet telephony.

As we can conclude from the above, two levels of congestion control a
required, one that operates at the global internetwork level and the oth
that operates at the router level. The aim of the first is to limit the aggrega
rate at which packets are entering the global internetwork to below its ma
mum rate, and the aim of the second is to maximize the flow of pack
through each router. In the following subsections we discuss aspects of two
the schemes that are used to perform these functions.

9.8.1 Integrated services

Most early applications of the Internet were text based and hence relativ
insensitive to delay and jitter. Examples include FTP and email, both of wh
can tolerate the added delays incurred by the use of a host-to-host retransm
sion control mechanism to overcome the effect of lost packets resulting fr
the best-effort service provided by IP. Other text-based applications, howev
cannot tolerate the delay caused by retransmissions but nevertheless requ
minimal packet losses. Examples of this type of application are those relati
to network control.

More recently, a number of interpersonal applications involving pack
tized speech and video were introduced. These require the packets that a
generated at the source to be transferred over the Internet and played out a
the destination in real time. This means that the retransmission of lost pac

ets is not possible and that the packet flow is particularly sensitive to lost packets and jitter. Such applications also require a guaranteed minimum bandwidth. To meet this more varied set of QoS requirements, two schemes have been researched and standardized, one called integrated services (IntServ) and the other differentiated services (DiffServ). In this section we describe aspects of the IntServ scheme while aspects of DiffServ are described in Section 9.8.2.

In both schemes, the packets relating to the different types of call/session are each allocated a different value in the precedence bits of the *type of service* field of the IP packet header. This is used by the routers within the Internet to differentiate between the packet flows relating to the different types of call. The IntServ solution defines three different classes of service:

- guaranteed: in this class, a specified maximum delay and jitter and an assured level of bandwidth are guaranteed. It is intended for applications involving the playout of real-time streams;
- controlled load (also known as predictive): in this class, no firm guarantees are provided but the flow obtains a constant level of service equivalent to that obtained with the best-effort service at light loads. Examples of applications in this class are those involving real-time streams that have the capability of adjusting the amount of real-time data that is generated to the level that is offered;
- best-effort: this is intended for text-based applications.

To cater for the three different types of packet flows, within each router, three separate output queues are used for each line, one for each class. In addition, appropriate control mechanisms are used to ensure the QoS requirements of each class are met. We shall first discuss a number of these as these are used in both the IntServ and the DiffServ schemes.

Token bucket filter

This is used with each of the packet flows in both the guaranteed and predictive service classes. A portion of the bandwidth of the outgoing line and an amount of buffer/queue space is reserved for the packet flow relating to each call. A control mechanism called the token bucket filter is used to enforce these allocations so that the guaranteed QoS requirements in terms of bandwidth, delay, and jitter are met.

Associated with each flow is a container called a *bucket* into which tokens are entered at a rate determined by the bandwidth requirement of the flow. The size of the bucket is the same as the maximum amount of buffer/queue space the flow may consume. A packet relating to a flow can only be transferred to the output queue if there are sufficient tokens currently in the bucket. The number of tokens required is determined by the packet length and, if sufficient tokens are currently in the bucket, the packet is queued and the corresponding number of tokens taken from the bucket. As we can deduce from this, therefore, providing the arrival rate of packets is less than or equa to the rate of entry of tokens into the bucket, then both the agreed bandwidth and delay/jitter will be met. If the arrival rate of packets exceeds th allocated bandwidth, normally these are relegated to the best-effort queue.

Weighted fair queuing

Since the packet rate – and hence bandwidth – associated with each flow may be different, when the packets relating to each flow are queued for transmission, in order to ensure the guaranteed delay bounds are met, it is necessary to ensure that the packets relating to each flow are not delayed by the packets of other flows. Hence a queue management scheme is also required to schedule the order that queued packets are transmitted. The **weighted fair queuing (WFQ)** scheme performs this function.

In order to ensure the delay bounds for each flow are met, the order of transmission of packets from the queue is changed each time a new packet arrives for queuing. When a packet arrives at an incoming line of the router it is given a time-stamp. This is determined from the arrival time of the packet and its scheduled departure time, the latter computed from the bandwidth associated with the flow and the packet length. The time-stamp of the packet is then compared with the time-stamps of the packets that are currently queued and the packet with the smallest time-stamp is transmitted first. In this way the delay bounds of each flow are met.

Random early detection

The requirements of the queue management scheme used with the best-effort queue are different from those of the other two queues. As we indicated earlier, normally, a router simply discards/drops a packet if the required output queue is already full. However, as we shall describe later in Section 12.3.2, with TCP, each time a packet relating to a call/session is lost, the TCP in the source host detects this and halves its current rate of entry of new packets for the call. Since this is done by all the hosts that lose a packet, the utilization of the link bandwidth falls dramatically. This is also detected by the affected TCPs which then quickly ramp-up the rate of entry of new packets. This, in turn, often results in full queues and dropped packets occurring again with the effect that the utilization of the available transmission bandwidth is poor. To stop this occurring, the **random early detection (RED)** queue management scheme is often used.

With RED, when a packet arrives for an output queue and the queue is full, instead of discarding the packet, a packet that is already in the queue is randomly selected for discarding. This has the affect that a reduced number of different applications are affected and hence the bandwidth utilization of the link is much improved.

To implement the scheme, two thresholds relating to the queue are defined: a minimum threshold (MinTH) and a maximum threshold (MaxTH). Also, the average length (AvrLEN) of the queue is continuously

monitored and used as a measure of the current level of traffic using the line. The action taken by the scheduler is determined by the current AvrLEN relative to the two thresholds as follows:

AvrLEN < MinTH:	the new packet is entered into the queue;
AvrLEN < MaxTH:	the new packet is dropped;
MaxTH < AvrLEN < MinTH:	a randomly-selected packet from the queue is dropped and the new packet is queued.

As we can deduce from the last condition, the probability of packets that are already in the queue being dropped increases as the AvrLEN increases. This has been found to give high levels of bandwidth utilization during periods of congestion.

Resource reservation protocol

With the IntServ scheme, in order to ensure the aggregate bandwidth of real-time traffic flows does not exceed that which is allocated for both the guaranteed and controlled-load traffic, the resources required for each flow (in terms of transmission bandwidth and buffer capacity) are reserved in advance of each packet flow starting. The protocol used to do this is called the **resource reservation protocol (RSVP)**.

Because many of the new real-time applications involve multiple participants, RSVP is used to reserve resources in each router along either a unicast or a multicast path. The actual routing of the packets associated with both types of call, however, are not part of RSVP and these are carried out in the normal way using one of the routing algorithms we described earlier in Section 9.6. A selection of the messages associated with RSVP are shown in Figure 9.28(a).

Each traffic flow is identified uniquely by the combined source and destination addresses in the IP header and the port number in the UDP header. To perform a reservation, the AP in the host that wishes to set up the call sends a *path* message in a UDP datagram with either the unicast or multicast address of the other host(s) in the destination address field of the IP header and the port number in the UDP datagram header. The purpose of the *path* message is firstly, to enable each router along the path/route followed by the packet to create an entry in a table known as the **path-state table** and secondly, to gather information about the resources that are currently available in each router along the path. The entry in the path-state table includes the flow identifier, a specification of the required parameters associated with the call – known as the traffic specification or Tspec – and the (IP) address of the router from which it received the *path* message.

On receipt of the *path* message, the AP (identified by the port number) in the destination host(s) uses the resource levels reported by each router to determine which type of call – guaranteed, controlled-load, or neither – the

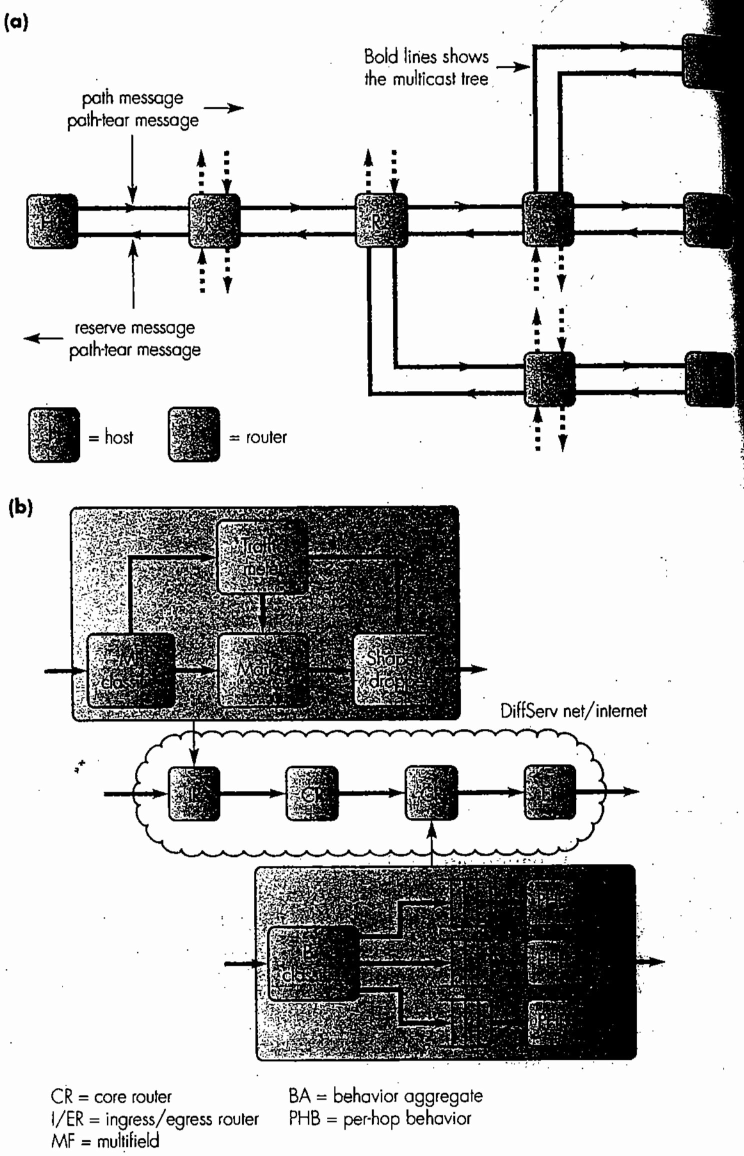


Figure 9.28 QoS support mechanisms: (a) RSVP principles; (b) DiffServ architecture.

path can support. Assuming the call can be accepted, the AP in each destination host then returns a *reserve* message (containing the Tspec for the call) to the router from which it received the *path* message. If the router still has the necessary resources, it reserves these for the flow (the resources may be different in each direction), makes an entry in the path-state table of the address of the router which sent it the message, and then sends the (*reserve*) message to the next router along the path. If the resources are not available, then a *path-tear* message is returned along the forward and return paths in order to release any resources that have been reserved and delete the entry in the path-state table. This procedure is repeated by each router along the path back to the source which, on receipt of the *reserve* message, proceeds with the call.

The path associated with each call/session may change during the lifetime of a call; for example, due to a router going down. To allow for this occurring, associated with each entry in the path-state table kept by each router is a timer called the *cleanup timer* and, should this expire, the entry is deleted. The timer is restarted each time a *path* message relating to the call is received. At periodic intervals – less than the cleanup timeout period – the source host sends a new *path* message which is acted upon by each router in the same way as the first *path* message. Hence should the new path not include a particular router, its cleanup timer will expire and the related path-state information be deleted. This mode of operation is known therefore as *soft-state* since it may change during a call.

Finally, on completion of the call/session, the AP that set up the call sends out a *path-tear* message which results in the entry in the path-table held by each router along the path(s) being deleted. The RFCs associated with RSVP and the other IntServ procedures are in **RFC 2205–2216**.

The major disadvantage of RSVP is that state information is retained by each router for each call/flow. Although this may be acceptable in a company intranet, for example, in a backbone router of the Internet the tables used for this purpose can be very large. Hence with the very high bit rates that are used, the heavy overheads per call can be unacceptably high. It is for this reason that the alternative DiffServ scheme was developed.

9.8.2 Differentiated services

Using the DiffServ approach, individual flows are not identified and instead the individual flows in each service class are aggregated together. Flows are then treated on a per-class basis rather than a per-flow basis. The general architecture used with a DiffServ network is shown in Figure 9.28(b).

The incoming packet flows relating to individual calls – referred to as microflows – are classified by the router/gateway at the edge of the DiffServ-compliant net/internet into one of the defined service/traffic classes by examining selected fields in the various headers in the packet. Within the DiffServ network the *type of service* (*TOS*) field in the IP packet header is replaced by a new field called the *differentiated services* (*DS*) *field.* As we saw ea lier in Section 9.2, this is an 8-bit field although currently only six bits ar used for the DS field. The six bits form what is called the **(DS) packet cod point (DSCP)** which is used to enable each router to determine the traf class to which the packet data relates and the output queue into which th packet should be put. The queue management/scheduling procedure rela ing to each queue is called a **per-hop behavior (PHB)** and, since this appli to an aggregated set of packet flows, a PHB is said to be applied to a **behavio aggregate** (**BA**).

Within the DiffServ network, a defined level of resources in terms of th buffer space within each router and the bandwidth of each output line is allo cated to each traffic class using, for example, a token bucket filter. As each packet arrives at the ingress router it is first classified as belonging to a parti ular traffic class. In some instances, however, as we shall see later, the actua classification is also a function of how well the microflow (to which the packe relates) is conforming to the agreed traffic profile for the flow. This is dete mined by the **traffic meter module** and, based on the level of conformance the traffic meter informs the **marker module** whether the packet should b marked with a low, medium, or high drop precedence. In addition, the traffi meter also informs the **shaper/dropper** module of this and the latter decide whether the packet should be dropped or allowed into the network.

Normally, real-time microflows with hard QoS guarantees are placed i the highest-priority traffic class. Typically, the traffic meter for this type o stream is a token bucket filter with a defined rate and bucket depth. Hence i an arriving packet relating to a flow in this class is deemed to be out-o profile, the traffic meter informs the shaper/dropper module of this. Th latter then either drops the packet or relegates it to the best-effort class by se ting all six DSCP bits to zero.

Once within the network, as each packet arrives at a core router, the **B classifier** first determines to which traffic class the packet belongs and hence to which output queue the packet should be transferred. Each queue is the serviced using an appropriate PHB. Currently, two PHBs have been defined **expedited forwarding** (**EF**) and **assured forwarding** (**AF**). The EF PHB is sim ilar to the guaranteed service class associated with IntServ and hence has th highest priority. The PHB used with this is based on a queue scheduling pr cedure such as weighted fair queuing.

The AF PHB has four ordered traffic classes associated with it each o which has three drop procedure levels: low, medium, and high. Should co gestion arise, this is used together with the traffic class to determine which packet(s) should be dropped. The PHB used in this case, therefore, is based on a queue scheduling procedure such as random early discard. The RFC associated with DiffServ are in **RFC 2474–5**.

9.9 The PPP link layer protocol

As we showed in Figure 9.1 and explained in the accompanying text, access networks are connected to the Internet global internetwork by gateways. Also, as we showed in Figure 9.9, access gateways are connected to interior gateways which, in turn, are interconnected together using high bit rate synchronous transmission lines leased from a network provider. Typically, the leased lines are either from the PDH – DS1/3 or E1/3 for example – or from the SDH/SONET hierarchy – STS-3/STM-1 for example. Hence, since the various types of gateway (router) operate at the IP (network) layer, as we showed in Figure 9.1, a link layer protocol is required to transfer the IP packets/datagrams over the different types of leased line.

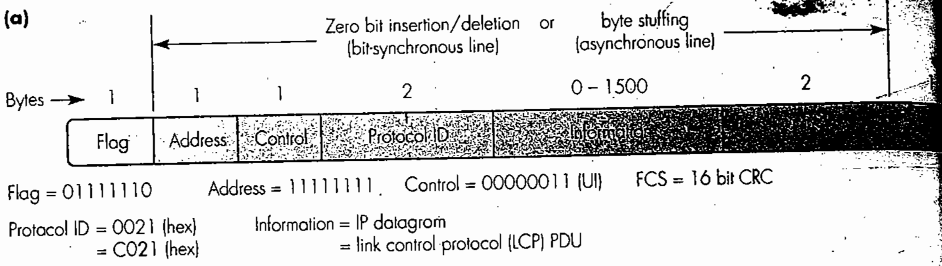
In addition, many of the access networks provided by ISPs require a link layer protocol to transfer the information entered by a person at home (using a PC for example) to the ISP network gateway. In many instances, the information is transferred over a switched connection through a PSTN using modems. Normally, the modem at home is connected to the serial port of the PC and, as we explained in Section 6.4, character-oriented asynchronous transmission is usually used. In order to avoid the proliferation of many different protocols, the IETF has defined a standard link layer protocol to meet these requirements. This is known as the **point-to-point protocol** (**PPP**) and is defined in **RFC 1661/2** and **3**.

To give the PPP the necessary flexibility, it has a number of features that enable it to be used in these and other applications. For example, it can operate in either the connection-oriented (reliable) or connectionless (best-effort) mode and with a variety of different types of network layer protocol. The latter feature is necessary, for example, if the access network uses the IPX network protocol rather than the IP. The PPP is based on the HDLC protocol we described in Section 6.8 and the format of all frames is shown in Figure 9.29.

Although the HDLC protocol is bit-oriented, in the PPP all frames are made up of an integral number of bytes/octets encapsulated by the opening and closing flag byte of 01111110. To achieve data transparency with the different types of synchronous transmission lines, zero bit insertion and deletion is used, the operation of which we described in Section 6.5.3.

In the case of asynchronous transmission lines, all characters are made equal to 8 bits and a form of character/byte stuffing is used, the principle of which we described in Section 6.4.3. In this application, however, in addition to flag bytes, a number of different characters/bytes are transmitted transparently. These include all the 32 control characters in the ASCII character set we identified in Figure 2.6(a). The list of bytes/characters and their codes is given in Figure 9.29(b).

As we can see, the escape byte/character used is 01111101 – 7D (hex) – and, whenever this is inserted, the sixth bit of the following byte/character is complemented. This rule is also used to transfer a byte/character that is equal to the escape byte.



(b)

Byte/character	Normal encoding	Byte-stuffed encoding	
Flag	01111110	01111101	01011110
NUL	00000000	01111101	00100000
SOH	00000001	01111101	00100001
⋮			
SI	00001111	01111101	00101111
US	00011111	01111101	00111111
Escape	01111101	01111101	01011101

Figure 9.29 The point-to-point protocol (PPP): (a) frame format; (b) byte stuffing princip

Since the PPP is intended for use over point-to-point lines, the *add* field has no role to play. Hence it is always set to all binary 1s. Also, sinc most applications all information is transferred over the line in the con tionless mode, as we explained in Section 6.8, the default value in the *co* field is 03 (hex) which is the code used to indicate an unnumbered infor tion (UI) frame.

The default length of the *protocol ID* field is two bytes. It is used to indi the type of packet – and hence network layer protocol – that is present in *information* field. For example, a value of 0021 (hex) is used to indicate a packet/datagram is present. In addition, since the PPP can be used in a ety of applications, a single request-response protocol known as the **control protocol (LCP)** has been defined to allow the operational mode associated parameters for the line/link to be negotiated. Associated with LCP are a number of messages – protocol data units (PDUs) – and, when is present in the information field, the value in the *protocol ID* field is (hex). For example, when transferring only IP datagrams over a line, an *request* message can be sent to propose the use of a reduced frame hea with no address or control fields and just a single byte for the protocol fi The other side then responds by returning either an *accept* message or a message in which case the default values must be used.

The default maximum size of the information field is 1500 by although a different maximum size can be negotiated using LCP reque

---

response messages. The *FCS* field is used to detect the presence of transmission errors in the frame and, as we explained in Section 6.8, it is based on a CRC. The FCS has a default length of 16 bits but a length of 32 bits can be negotiated using the LCP.

9.10 IPv6

Until the mid-1990s the Internet was used primarily by universities, government agencies, research establishments, and some sectors of industry. Since that time, however, it has gone through unprecedented growth owing to a large extent to the rapid rise in interest in the use of the World Wide Web. As a result, most schools, colleges, and many homes now have PCs with connections to the Internet. These are now being used, in addition to Web access, to exploit the range of other applications that are supported by the Internet. Moreover, this growth is predicted to increase even faster as new applications emerge: for example, the widespread use of hybrid mobile phones/computers with Internet interfaces, television set-top boxes with integral Web browsers, plus a potentially vast number of consumer products – such as meters, household appliances, office equipment, and so on – many of which may have an Internet interface.

Although the introduction of CIDR has extended considerably the usable address space of IPv4, the IETF has already defined and is using a new version of IP which has a number of features that have been introduced to meet the predicted growth levels. This is known as **IP version 6 (IPv6)** or sometimes **IP next generation (IPng)**. It is defined in **RFCs 1883–7** and a number of supporting RFCs. In addition to providing a large increase in the number of IP addresses, the IETF has taken the opportunity to correct some of the deficiencies associated with IPv4 and to provide a number of other features. The main new features of IPv6 are:

- a much-increased address space from 32 bits to 128 bits;
- hierarchical addresses to reduce the size of the routing tables associated with the routers in the core backbone network;
- a simplified header to enable routers and gateways to process and route packets faster;
- the introduction of improved security and data integrity features including authentication and encryption;
- an autoconfiguration facility that enables a host to obtain an IP address via the network without human intervention;
- harder quality-of-service guarantees by means of the preferential treatment by routers of the packets associated with interactive and multimedia applications relative to those relating to traditional applications such as email and file transfers;

- support for mobile computing by the use of autoconfiguration to obtain an IP address dynamically via the network for the duration of a call/session.

Although many of the above features require some radical changes t IPv4 – for example a different address structure and datagram/packet forma – in terms of the protocols that are used within the expanded global internet work, these operate in much the same way as the current IPv4 protocols. Fo example, the LS-SPF (OSPF) routing algorithm we described in Section 9.6.4 is used as the standard interior gateway protocol (IGP) for IPv6 except, of course, that 128-bit addresses are used in the link-state and routing tables There is also an updated version of the RIP – based on the distance vecto routing algorithm we described in Section 9.6.3 – called **RIPng**. Similarly, a the backbone level, the border gateway protocol (BGP) we described in Section 9.6.5 (including the CIDR we described in Section 9.6.6) is used bu with extensions to allow reachability information based on IPv6 hierarchical addresses to be exchanged. Hence in the remainder of this section we limi our discussion to a selection of the new features that are used.

9.10.1 Datagram format

In relation to the IPv4 datagram/packet header, a number of fields have been dropped and others have been made optional. The result is a basic/main header of 40 bytes, the contents and format of which are shown in Figure 9.30(a). The use of each field is as follows:

Version

This is set to 6 to enable routers to discriminate IPv6 packets from IPv4 packets. It is envisaged that both will need to coexist for many years.

Traffic class

This field plays a similar role to the *ToS* byte in the IPv4 header. It allows th source IP to allocate a different priority to packets relating to, say, multimedia applications involving real-time streams from those relating to traditional applications. It contains a 4-bit priority field and hence 16 priorities are possible, th higher the priority value the higher the packet priority. Values in the range 0– are for packets relating to applications for which best-effort delivery is acceptable; for example network news (1) and FTP (4). Both these applications ar less sensitive to delay and delay variation (jitter) than applications involving real-time media but FTP is more sensitive to packet loss than network news Values in the range 8–15 are for packets containing real-time streams such as audio and video. Typically, such streams/packets are more sensitive to delay and jitter than the packets in the first category and hence should be transmitted before them during periods of congestion. Also, within the second category, packets containing compressed video are more sensitive to packet loss than packets containing just audio and hence are given a higher priority.

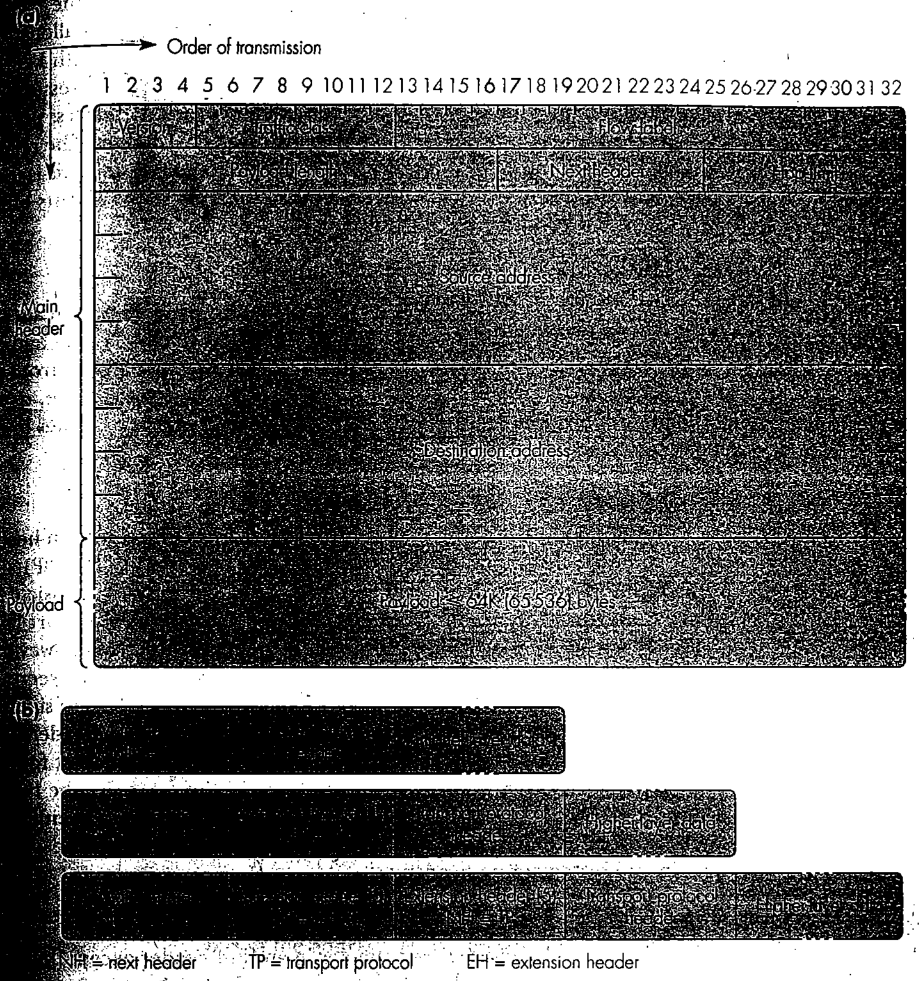

(a)
Order of transmission
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Version
Traffic class
Flow label
Payload length
Next header
Hop limit
Source address
Destination address
Main header
Payload
Payload ≤ 64K (65 536) bytes
(b)
Higher-layer data
Transport protocol header
Higher-layer data
Extension header (EH) NH = TP
Transport protocol header
Higher-layer data
NH = next header
TP = transport protocol
EH = extension header

Figure 9.30 IPv6: (a) main header fields and format; (b) position and order of extension headers.

Flow label

This is a new field and is closely linked to the *traffic class* field. It is set to zero in best-effort packets and, in packets in the second category, it is used to enable a router to identify the individual packets relating to the same call/session. As we saw earlier in Section 9.8.1, one approach to handling packets containing real-time streams is to reserve resources – for example transmission bandwidth – for such calls in advance of sending the pac relating to the call. During the reservation procedure, the call is allocat *flow label* by the source. Also, each router along the reserved path kee record of this, together with the source and destination IP addresses, table. Routers then use the combined *flow label* and source IP address pre in each packet header to relate the packet to a specific call/flow. The rel routing information is then retrieved from the routing table and this is together with the *traffic class* value, to ensure the QoS requirements of call/flow are met during the forwarding procedure.

Payload length

This indicates the number of bytes that follow the basic 40-byte heade the datagram. The minimum length of a basic datagram is 536 bytes and maximum length is 64K bytes. The *payload length* is slightly different from *total length* field used in the header of an IPv4 datagram since, as explained in Section 9.2, the *total length* includes the number of bytes in datagram header.

Next header

As we show in Figure 9.30(b), a basic IPv6 datagram comprises a main he followed by the header of the peer transport layer protocol (TCP/UDP) a where appropriate, the data relating to the higher layers. With a basic gram, therefore, the *next header* field indicates the type of transport protocol (header) that follows the basic header. If required, howeve number of what are called ***extension headers*** can be inserted between main header and the transport protocol header. Currently, there are six of extension header defined and, when present, each extension header with a new *next header* field which indicates the type of header that fol The *next header* field in the last extension header always indicates the ty transport protocol header that follows. Thus, the *next header* field in eith main header or the last extension header plays the same role as the field in an IPv4 datagram header.

Hop limit

This is similar to the *time-to-live* parameter in an IPv4 header except the is a hop count instead of a time. In practice, as we explained in Sectio most IPv4 routers also use this field as a hop count so the change in the name simply reflects this. The initial value in the *hop limit* field is set source and is decremented by 1 each time the packet/datagram is forwa The packet is discarded if the value is decremented to zero.

Source address and destination address

As we indicated earlier, these are 128-bit addresses that are used to ide the source of the datagram and the intended recipient. In most cases this be the destination host but, as we shall explain later, it might be the

---

router along a path if source routing is being used. Unlike IPv4 addresses, an IPv6 address is assigned to the (physical) interface, not to the host or router. Hence in the case of routers (which have multiple interfaces) these are identified using any of the assigned interface addresses.

9.10.2 Address structure

As we showed in Figure 9.9, the various networks and internetworks that make up the global Internet are interconnected in a hierarchical way with the access networks at the lowest level in the hierarchy and the global backbone network at the highest level. However, the lack of structure in the netid part of IPv4 addresses means that the number of entries in the routing tables held by each gateway/router increases with increasing height in the hierarchy. At the lowest level, most access gateways associated with a single site LAN have a single netid while at the highest level, most backbone routers/gateways have a routing table containing many thousands of netids.

In contrast, the addresses associated with telephone networks are hierarchical with, for example, a country, region, and exchange code preceding the local number. This has a significant impact on the size of the routing tables held by the switches since, at a particular level, all calls with the same preceding code are routed in the same way. This is known as **address aggregation.**

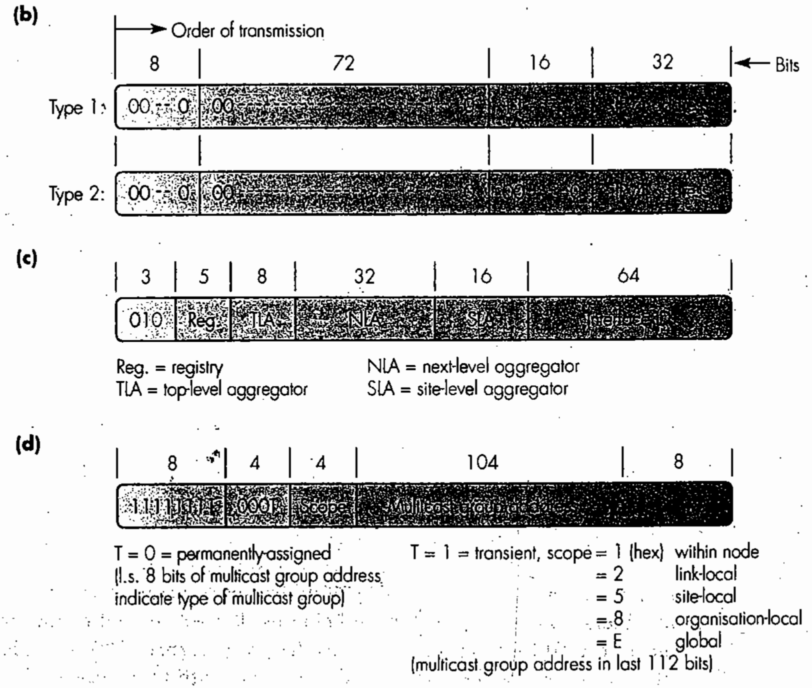
As we explained earlier in Section 9.6.6, classless inter-domain routing is a way of introducing a similar structure with IPv4 addresses and reduces considerably the size of the routing tables held by the routers/gateways in the global backbone. From the outset, therefore, IPv6 addresses are hierarchical. Unlike telephone numbers, however, the hierarchy is not constrained just to a geographical breakdown. The large address space available means that a number of alternative formats can be used. For example, to help interworking with existing IPv4 hosts and routers, there is a format that allows IPv4 addresses to be embedded into an IPv6 address. Also, since the majority of access networks are now Internet service provider (ISP) networks, there is a format that allows large blocks of addresses to be allocated to individual providers. The particular format being used is determined by the first set of bits in the address. This is known as the **prefix format (PF)** and a list of the prefixes that have been assigned – together with their usage – is given in Figure 9.31(a).

Unicast addresses

As we can see, addresses starting with a prefix of 0000 0000 are used to carry existing IPv4 addresses. There are two types, the formats of which are shown in Figure 9.31(b). As we shall expand upon in Section 9.11, a common requirement during the transition from IPv4 to IPv6 is to tunnel the IPv6 packets being generated by the two communicating IPv6 hosts – often written **V6 hosts** – over an existing IPv4 network/internetwork. Hence to simplify the routing of the IPv4 packet – containing the IPv6 packet within it – the IPv6

(a)

Prefix format	Usage
0000 0000	Embedded IPv4 address
0000 001	Embedded OSI address
0000 010	Embedded Novell NetWare IPX address
010	Provider-based unicast address
100	Geographic-based unicast address
1111 1110 10	Link local-use address
1111 1110 11	Site local-use address
1111 1111	Multicast address


(b)
Order of transmission
8
72
16
32
Bits
Type 1:
00 - - 0
00
Type 2:
00 - - 0
00
(c)
3
5
8
32
16
64
010
Reg.
TLA
NLA
SLA
Interface ID
Reg. = registry
TLA = top-level aggregator
NLA = next-level aggregator
SLA = site-level aggregator
(d)
8
4
4
104
8
1111 1111
000T
Scope
Multicast group address
T = 0 = permanently-assigned
(l.s. 8 bits of multicast group address indicate type of multicast group)
T = 1 = transient, scope = 1 (hex) within node
= 2 link-local
= 5 site-local
= 8 organisation-local
= E global
(multicast group address in last 112 bits)

ire 9.31 IPv6 addresses: (a) prefix formats and their use; (b) IPv4 address types; provider-based unicast address format; (d) multicast address format.

address contains the IPv4 address of the destination gateway embedded within it. The second type is to enable a V4 host to communicate with a V6 host. The IPv4 address of the V4 host is then preceded by 96 zeros. In addition, two addresses in this category are reserved for other uses. An address comprising all zeros indicates there is no address present. As we shall expand upon in Section 9.10.4, an example of its use is for the source address in a packet relating to the autoconfiguration procedure. An address with a single binary 1 in the least significant bit position is reserved for the loopback address used during the test procedure of a host protocol stack.

The OSI and NetWare address prefixes have been defined to enable a host connected to one of these networks to communicate directly with a V6 host. The most widely used is the provider-based prefix as this format reflects the current structure of the Internet. A typical format of this type of address is shown in Figure 9.31(c). As we saw in Figure 9.9, the core backbone of the Internet consists of a number of very high bandwidth lines that interconnect the various continental backbones together. The routers that perform this function are owned by companies known as **top-level aggregators (TLAs)**. Each TLA is allocated a large block of addresses by what is called a **registry**, the identity of which is in the field immediately following the 010 prefix. Examples include the North American registry, the European registry, the Asia and Pacific registry, and so on.

From their allocation, the TLAs allocate blocks of addresses to large Internet service providers and global enterprises. These are known as **next-level aggregators (NLAs)** and, in the context of Figure 9.9, operate at the continental backbone and national and regional levels. The various NLAs allocate both single addresses to individual subscribers and blocks of addresses to large business customers. The latter are known as **site-level aggregators (SLAs)** and include ISPs that operate at the regional and national levels. The 64-bit **interface ID** is divided locally into a fixed subnetid part and a hostid part. Typically, the latter is the 48-bit MAC address of the host and hence 16 bits are available for subnetting.

As we can deduce from Figure 9.31(c), the use of hierarchical addresses means that each router in the hierarchy can quickly determine whether a packet should be routed to a higher-level router or to another router at the same level simply by examining the appropriate prefix. Also, the routers at each level can route packets directly using the related prefix. The same overall description applies to the processing of geographic-based addresses.

As their names imply, the two types of **local-use addresses** are for local use only and have no meaning in the context of the global Internet. As we shall expand upon in Section 9.10.4, *link local-use addresses* are used in the autoconfiguration procedure followed by hosts to obtain an IPv6 address from a local router. The router only replies to the host on the same link the request was received and hence this type of packet is not forwarded beyond the router.

The *site local-use addresses* are used, for example, by organizations that are not currently connected to the Internet but wish to utilize the technology associated with it. Normally, the 64-bit interface ID part is subdivided and used for routing purposes within the organization. In this way, should the organization wish to be connected to the Internet at a later date, it is only necessary to change the site local-use prefix with the allocated subscriber prefix.

Multicast addresses

The format of an IPv6 multicast address is shown in Figure 9.31(d). As we c
see, following the multicast prefix are two additional fields which have b
introduced to limit the geographic scope of the related multicast operatio
The first is known as the *flags* field and is used to indicate whether the mu
cast is a permanently-assigned (reserved) address (0000) or a tempora
(transient) address (0001). In the case of a permanently-assigned address, t
least significant 8 bits of the *multicast group address* field identify the type
the multicast operation, while for a transient address, the full 112 bits ident
the multicast group address.

In both cases, the 4-bit *scope* field defines the geographic scope of t
multicast packet. The various alternatives are identified in the figure a
mrouters use this field to determine whether the (multicast) packet shou
be forwarded further or discarded.

Anycast addresses

In addition to unicast and multicast addresses, a new address type known
an ***anycast group address*** has been defined. These are allocated from th
unicast address space and are indistinguishable from a unicast address. Wi
an anycast address, however, a group of hosts or routers can all have the sam
(anycast) address. A common requirement in a number of applications is fo
a host or router to send a packet to any one of a group of hosts or routers, a
of which provide the same service. For example, a group of servers distri
uted around a network may all contain the same database. Hence in order t
avoid all clients needing to know the unique address of its nearest server, a
the servers can be members of the same anycast group and hence have t
same address. In this way, when a client makes a request, it uses the assigne
anycast address of the group and the request will automatically be received
its nearest server. Similarly, if a single network/internetwork has a number
gateway routers associated with it, they can all be allocated the same anyca
address. As a result, the shortest-path routes from all other networks/inte
networks will automatically use the gateway nearest to them.

In order to perform the routing function, although an anycast addr
has the same format as a unicast address, when an anycast address is assign
to a group of hosts or routers, it is necessary for each host/router to
explicitly informed – by network management for example – that it i
member of an anycast group. In addition, each is informed of the comm
part of the address prefixes which collectively identify the topological regio
in which all the hosts/routers reside. Within this region, all the routers the
maintain a separate entry for each member of the group in its routing table.

Address representation

A different form of representation of IPv6 addresses has been defined
Instead of each 8-bit group being represented as a decimal number (with
dot/period between them), groups of 16 bits are used. Each 16-bit group i

then represented in its hexadecimal form with a colon between each group. An example of an IPv6 address is:

FEDC:BA98:7654:3210:0000:0000:0000:0089

In addition, a number of abbreviations have been defined:

- One or more consecutive groups of all zeros can be replaced by a pair of colons.
- Leading zeros in a group can be omitted.

Hence the preceding address can also be written as:

FEDC:BA98:7654:3210::89

Also, for the two IPv4 embedded address types, the actual IPv4 address can remain in its dotted decimal form. Hence assuming a dotted decimal address of 15.10.0.6, the two embedded forms are:

:: 150.10.0.6	(IPv4 host address)
:: FFFF:150.10.0.6	(IPv4 tunnel address)

Example 9.6

Derive the hexadecimal form of representation of the following link-local multicast addresses:

(i) a permanently-assigned multicast group address of 67
(ii) a transient multicast group address of 317

Answer:

The formats of the two types of multicast addresses were shown in Figure 9.31(d). Hence:

The most significant 16 bits are FF02 = permanently assigned, link-local
and FF12 = transient, link-local

(i) A permanently-assigned multicast group address of 67 = 0043 (hex)
Hence IPv6 address = FF02::[illegible]

(ii) A transient multicast group address of 317 = 013D (hex)
Hence IPv6 address = FF12::13D

9.10.3 Extension headers

As we have just indicated, if required, a number of extension headers can added to the main header to convey additional information, either to routers visited along the path followed or to the destination host. The types of extension header currently defined are:

- **hop-by-hop options**: information for the routers visited along a path;
- **routing**: list of routers relating to source routing information;
- **fragment**: information to enable the destination to reassemble a fragmented message:
- **authentication**: information to enable the destination to verify the identity of the source;
- **encapsulating security payload**: information to enable the destination to decrypt the payload contents;
- **destination options**: optional information for use by the destination.

The two options headers can contain a variable number of option fields each possibly of a variable length. For these, each option field is encoded usin a **type-length-value** (TLV) format. The *type* and *length* are both single bytes an indicate the option type and its length (in bytes) respectively. The *value* is the found in the following number of bytes indicated by the *length*. The option typ identifiers have been chosen so that the most significant two bits specify th action that must be taken if the type is not recognized. These are:

00 ignore this option and continue processing the other option fields in the header;
01 discard the complete packet;
10 discard the packet and return an ICMP error report to the source indicating a parameter problem and the option type not recognized
11 same as for 10 except the ICMP report is only returned if the destination address is not a multicast address.

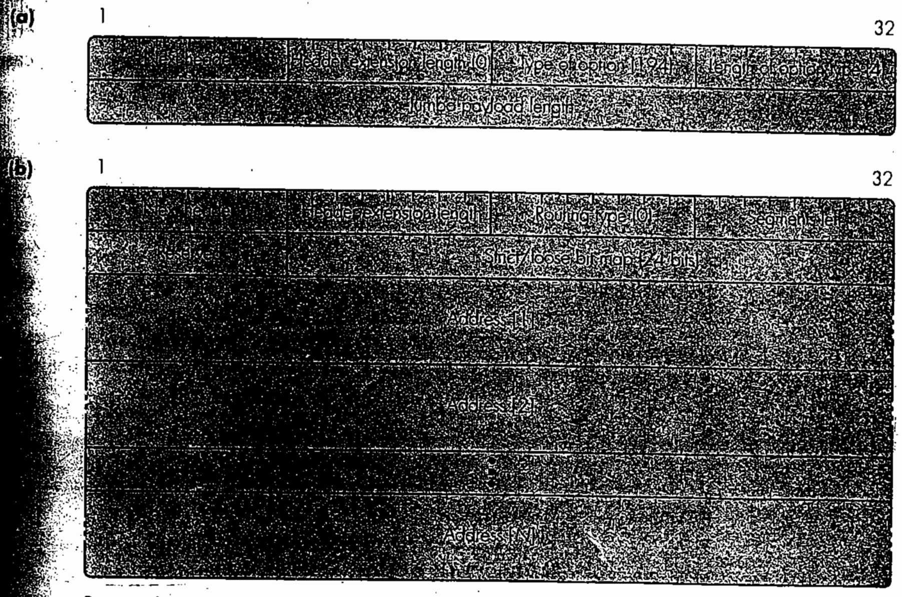
Note also that since the type of extension header is indicated in the pr ceding *next header* field, the related decoder that has been written to deco the contents of the header is invoked automatically as each header processed. Some examples of each header type now follow.

Hop-by-hop options

This type of header contains information that must be examined by all th gateways and routers the packet visits along its route. The *next header* value fo this is 0 and, if this header is present, it must follow the main header. Hence the *next header* field in the main header is set to 0. An example of its use is for a host to send a datagram that contains a payload of more than 64K bytes.

This is particularly useful, for example, when a host is transferring many very large files over a path that supports a maximum transmission unit (MTU) significantly greater than 64K bytes. Datagrams that contain this header are known as **jumbograms** and the format of the header is shown in Figure 9.32(a).

As we can see, this type of header is of fixed length and comprises two 32-bit words (8 bytes). The *header extension length* field indicates the length of the header in multiples of 8 bytes, excluding the first 8 bytes. Hence in this case the field is 0. This option contains only one option field, the *jumbo payload length*. As we indicated earlier, this is encoded in the TLV format. The *type* for this option is 194 (11000010) and the *length* is 4 (bytes). The *value* in the *jumbo payload length* is the length of the packet in bytes, excluding the main header but including the 8 bytes in the extension header. This makes the *payload length* in the main header redundant and hence this is set to zero.



Bit map: 1 = strict source routing
0 = loose source routing

$N \leqslant 24$

Next header: hop-by-hop options = 0
routing = 43

ure 9.32 **Extension header formats: (a) hop-by-hop options; (b) routing.**

Routing

This plays a similar role to the (strict) *source routing* and *loose source rou* optional headers used in IPv4 datagrams. The *next header* value for this typ header is 43. Currently only one type of routing header has been defined, format of which is shown in Figure 9.32(b).

The *header extension length* is the length of the header in multiples of bytes, excluding the first 8 bytes. Hence, as we can deduce from the figu this is equal to two times the number of (16-byte) addresses present in header. This must be an even number less than or equal to 46. The *segm* *left* field is an index to the list of addresses that are present. It is initialized 0 and is then incremented by the next router in the list as it is visited. T maximum therefore is 23.

The second 4-byte word contains a *reserved* byte followed by a 24-bit fi called the *strict/loose bit map*. This contains one bit for each potential addr present starting at the leftmost bit. If the related bit is a 1, then the addr must be that of a directly-attached (neighbor) router – that is, strict sou routing. If the bit is a 0, then the address is that of a router with possibly s eral other routers in between – loose source routing. The latter is used, example, when tunneling is required to forward the datagram. In the case

Example 9.7

A datagram is to be sent from a source host with an [illegible]
a destination host with an IPv6 address of B via a [illegible]
IPv6 routers. Assuming the addresses of the [illegible]
and R3 and strict source routing is to be used, [illegible]
tents of the initial values in the various fields in [illegible]
will be and (ii) list the contents of the source [illegible]
fields in the main header and the [illegible]
header as the datagram travels along the defined [illegible]

Answer:

(i) Extension header initial contents:

Next header = Transport [illegible]
Header extension length = 2 × [illegible]
Routing type = 0
Segments left = 0
Strict/loose bit map = 11100000 [illegible]
List of addresses = R1, R2, R3 [illegible]

(ii) Contents of main header fields:

At source SA = A DA = R1 [illegible]
At R1 SA = A DA = R2 [illegible]
At R2 SA = A DA = R3 [illegible]

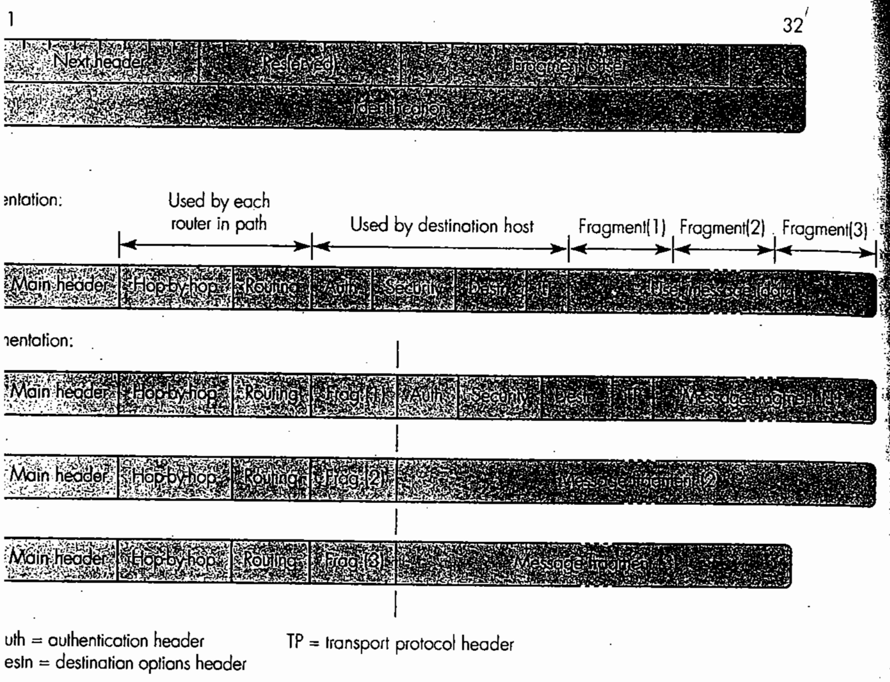
strict source routing, at each router the destination address in the main header is changed to that obtained from the list of addresses before the index is incremented. In the case of loose source routing, the destination address will be that of an attached neighbor which is on the shortest-path route to the specified address.

Fragment

This header is present if the original message submitted by the transport layer protocol exceeds the MTU of the path/route to be used. The *next header* value for a *fragment* extension header is 44. The fragmentation and reassembly procedures are similar to those used with IPv4 but, in the case of IPv6, the fragmentation procedure is carried out only in the source host and not by the routers/gateways along the path followed by the packet(s). As we saw in Figure 9.30(a), there is no don't fragment (D) bit in the IPv6 main header since, in order to speed up the processing/routing of packets, IPv6 routers do not support fragmentation. Hence, as we explained in Section 9.7, either the minimum MTU size of 576 bytes must be used or the *path MTU discovery* procedure is used to determine if the actual MTU size is greater than this. In either case, if the submitted message (including the transport protocol header) exceeds the chosen MTU (minus the 40 bytes for the IPv6 main header), then the message must be sent in multiple packets, each with a main header and a *fragment* extension header. The various fields and the format of each *fragment* extension header are shown in Figure 9.33(a). An example is shown in Figure 9.33(b).

Each packet contains a main header – plus, if required, a *hop-by-hop options* header and a *routing* header – followed by a *fragment* extension header and the fragment of the message being transmitted. Thus the maximum size of the payload (and hence message fragment) in each packet will be the MTU size being used minus the number of 8-byte fields required for the main header and any extension headers that are present. The *payload length* field in the main header of the first packet indicates the total number of bytes in the message being transmitted – including the IP header – plus the number of bytes that are required for the other extension headers that are being used. The *payload length* in the main header of the remaining packets indicates the number of bytes in the packet following the main header.

The various fields in each *fragment* header have similar functions to those used with IPv4. The *fragment offset* indicates the position of the first byte of the fragment contained within the packet relative to the start of the complete message being transmitted. Its value is in units of 8-bytes. The *M-bit* is the *more fragments bit*; it is a 1 if more fragments follow and a 0 if the packet contains the last fragment. Similarly, the value in the *identification* field is used by the destination host, together with the source address, to relate the data fragments contained within each packet to the same original message. Normally, the source uses a 32-bit counter (that is incremented by 1 for each new message transmitted) to keep track of the next value to use.


1
32
Next header
Reserved
Identification
entation:
Used by each router in path
Used by destination host
Fragment(1)
Fragment(2)
Fragment(3)
Main header
Hop-by-hop
Routing
Auth
Security
Destn
TP
User message data
entation:
Frag (1)
Message fragment(1)
Frag (2)
Message fragment(2)
Frag (3)
Message fragment(3)
uth = authentication header
estn = destination options header
TP = transport protocol header

33 IPv6 fragmentation: (a) fragment header fields and format; (b) example.

Authentication and encapsulating security payload

As we shall expand upon in Section 13.4, authentication and the related subject of encryption are both mechanisms that are used to enhance the security of a message during its transfer across a network. In the case of authentication, this enables the recipient of a message to validate that the message was indeed sent by the source address present in the packet/datagram header and not by an impostor. Encryption is concerned with ensuring that the contents of a message can only be read by – and hence have meaning to – the intended recipient. The *authentication* and *encapsulating* security payload (ESP) extension headers are present when both these features are being used at the network layer.

When using IPv6 authentication, prior to any information (packets) being exchanged, the two communicating hosts first use a secure algorithm to exchange secret keys. An example is the MD5 algorithm we describe later in Section 13.5. Then, for each direction of flow, the appropriate key is used to compute a checksum on the contents of the entire datagram/packet. The computed checksum is then carried in the authentication header of the packet. The same computation is repeated at the destination host and only if the computed checksum is the same as that carried in the authentication header is it acknowledged that the packet originated from the source host address indicated in the main header and also that the packet contents have not been modified during the packets transfer over the network. We discuss the subject of authentication further in Section 13.6.

The encryption algorithm used with the ESP is also based on the use of an agreed secret key. An example is the DES algorithm we describe later in Section 13.4.3. The agreed key is used to encrypt either the transport protocol header and payload parts of each packet or, in some instances, the entire packet including the main header and any extension headers present. In the case of the latter, the encrypted packet is then carried in a second packet containing a completely different main and, if necessary, extension headers. The first is known as **transport mode encryption** and the second **tunnel mode encryption.** Figure 9.34(a) illustrates the principle of operation of both modes.

As we can deduce from the figure, the overheads associated with the tunnel mode are significantly higher than those of the transport mode. The extra

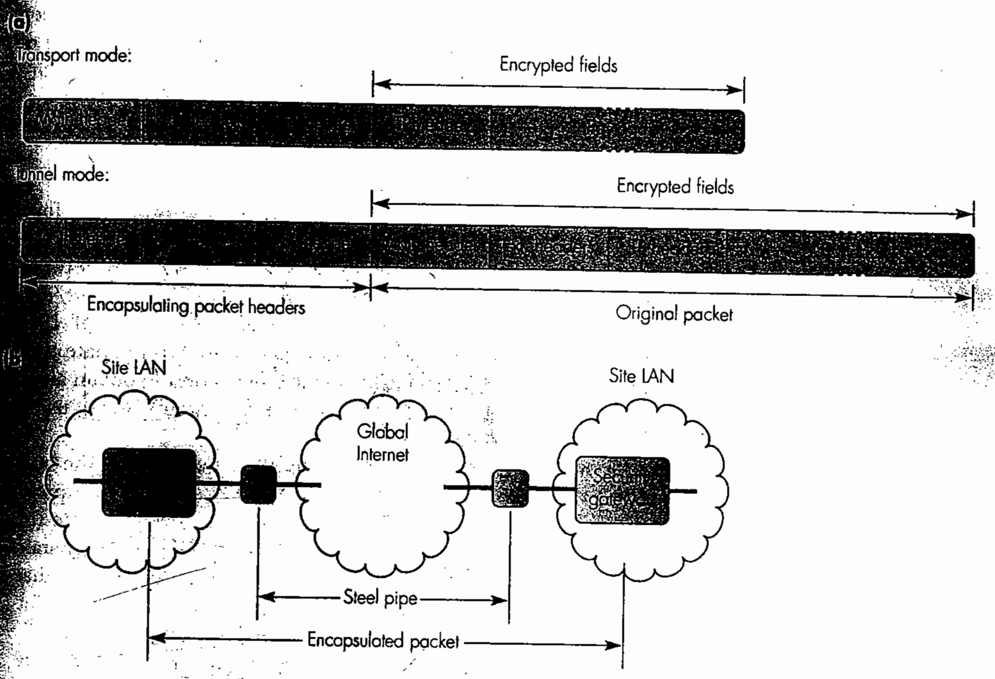

(a)
Transport mode:
Encrypted fields
Main header
Tunnel mode:
Encrypted fields
Encapsulating packet headers
Original packet
(b)
Site LAN
Site LAN
Global Internet
Security gateway
Steel pipe
Encapsulated packet

Figure 9.34 Encapsulating security payload: (a) transport and tunnel modes; (b) example application of tunnel mode.

security obtained with the tunnel mode is that the information in the main and extension headers of the original packet cannot be interpreted by a person passively monitoring the transmissions on a line. An example use of this mode is in multisite enterprise networks that use the (public) Internet to transfer packets from one site to another. The general scheme is shown in Figure 9.34(b).

As we showed earlier in Figure 5.16(a) and explained in the accompanying text, associated with each site is a security gateway through which all packet transfers to and from the site take place. Hence to ensure the header information (especially the routing header) of packets is not visible during the transfer of the packet across the Internet, the total packet is encrypted and inserted into a second packet by the IP in the security gateway with the IPv6 address of the two communicating gateways in the source and destination address fields of the main header. The path through the Internet connecting the two gateways is referred to as a **steel pipe**.

Destination options

These are used to convey information that is examined only by the destination host. As we indicated earlier, one of the ways of encoding options is to use the type-length-value format. Hence in order to ensure a header that uses this format comprises a multiple of 8 bytes, two *destination options* have been defined. These are known as Pad1 and Pad*N*, the first to insert one byte of padding and the second two or more bytes of padding. Currently these are the only two options defined.

9.10.4 Autoconfiguration

As we described earlier in Section 9.1, the allocation of the IP addresses for a new network involves a central authority to allocate a new netid – the Internet Network Information Center (InterNIC) – and a local network administrator to manage the allocation of hostids to each attached host/end system. Thus, the allocation, installation, and administration of IPv4 addresses can entail considerable effort and expenditure. To alleviate this IPv6 supports an autoconfiguration facility that enables a host to obtain an IP address dynamically via the network and, in the case of mobile hosts, use it just for the duration of the call/session.

Two types of autoconfiguration are supported. The first involves the host communicating with a local (site) router using a simple (stateless) request response protocol. The second involves the host communicating with a site (or enterprise) address server using an application protocol known as the **dynamic host configuration protocol (DHCP)**. The first is suitable for small networks that operate in the broadcast mode (such as an Ethernet LAN) and the second for larger networks in which the allocation of IP addresses needs to be managed.

With the first method, a simple protocol known as **neighbor discovery (ND)** is used. As we show in Figure 9.35, this involves the host broadcasting a *router solicitation* packet/message on the subnet/network and the router responding with a *router advertisement* message. Both messages are ICMPv6 messages and hence are carried in an IPv6 packet. The latter is then broadcast over the LAN in a standard frame.

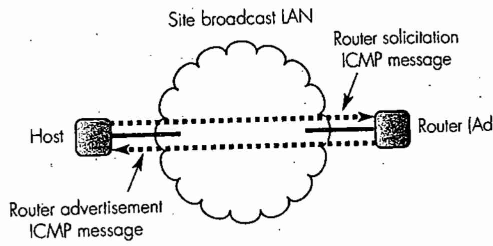

Site broadcast LAN
Router solicitation ICMP message
Host
Router (Address server)
Router advertisement ICMP message

Figure 9.35 Neighbor discovery protocol messages.

The main header of the IPv6 packet containing the router solicitation message has an IPv6 source address created by the host. This is made up of the (standard) link-local address prefix and the 48-bit MAC address of the host's LAN interface. As we indicated earlier, with IPv6 a number of permanent multicast group addresses have been defined including an all-routers group address. Hence the destination address in the packet main header is set to this and, since the packet is broadcast over the LAN, it is received by the ICMP in all the routers that are attached to the LAN.

A single router is selected to process this type of ICMP message and this responds to a router solicitation message with a route advertisement message containing the Internet-wide address prefix for the subnet/network. On receipt of this, the ICMP in the host proceeds to create its own IPv6 address by adding its 48-bit MAC address to the prefix. As we can deduce from this, in relation to IPv4, this is equivalent to the router providing the netid of the site and the host using its own MAC address as the hostid. Also, the same procedure can be used by mobile hosts.

With the second method, the host requests an IPv6 address from the site address server using the DHCP. The **DHCP address server** first validates the request and then allocates an address from the managed list of addresses the server contains. Alternatively if a site does not have its own address server – for example if the site LAN is part of a larger enterprise network – then a designated router acts as a **DHCP relay agent** to forward the request to the DHCP address server.

9.11 IPv6/IPv4 interoperability

The widespread deployment of IPv4 equipment means that the introduction of IPv6 is being carried out in an incremental way. Hence a substantial amount of the ongoing standardization effort associated with IPv6 is concerned with the interoperability of the newer IPv6 equipment with existing

IPv4 equipment. Normally, when a new network/internetwork is created, it is based on the IPv6 protocol and, in the context of the existing (IPv4) Internet, it is referred to as an **IPv6 island**. It is necessary to provide a means of interworking between the two types of network at both the address level and the protocol level. In this section, we identify a number of situations where interoperability is required and describe a selection of the techniques that are used to achieve this.

9.11.1 Dual protocols

Dual stacks are already widely used in networks that use dissimilar protocol stacks; for example a site server that supports IPX on one port and IP on a second port. In a similar way, dual protocols can be used to support both IPv4 and IPv6 concurrently. An example is shown in Figure 9.36.

In this example, the site has a mix of hosts, some that use IPv4 and others IPv6. In order to be able to respond to requests originating from both types of host, the server machine has both an IPv4 and an IPv6 protocol at the network layer. The value in the version field of the datagram header is then used by the link layer protocol to pass a datagram to the appropriate IP. In this way the upper layer protocols are unaware of the type of IP being used at the network layer.

9.11.2 Dual stacks and tunneling

A common requirement is to interconnect two IPv6 islands (networks/internetworks) through an intermediate IPv4 network/internetwork. To achieve this, the gateway/router that connects each IPv6 island to the IPv4 network must have dual stacks, one that supports IPv6 and the other IPv4. The IPv6 packets are then transferred over the IPv4 network using tunneling. The general approach is illustrated in Figure 9.37(a) and the protocols involved in Figure 9.37(b).

As we showed earlier in Figure 9.20 and explained in the accompanying text, tunneling is used to transfer a packet relating to one type of network layer protocol across a network that uses a different type of network layer protocol. Hence in the example shown in Figure 9.37, each IPv6 packet is transferred from one (IPv6/IPv4) edge gateway to the other edge gateway

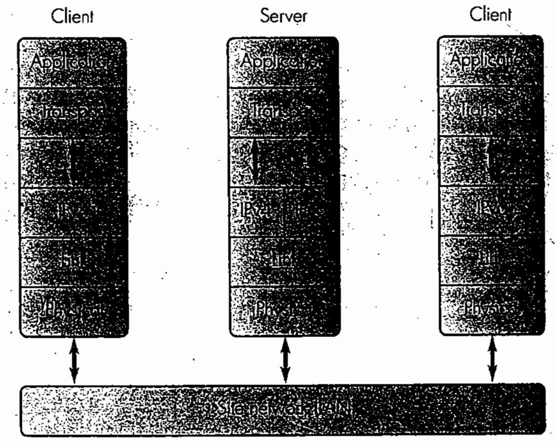

Client
Server
Client
Applicati
Applicati
Applicati
Transp
Trans
Trans
IPv4
Site network (LAN)

Figure 9.36 IPv6/IPv4 interoperability using dual (IPv6/IPv4) protocols.

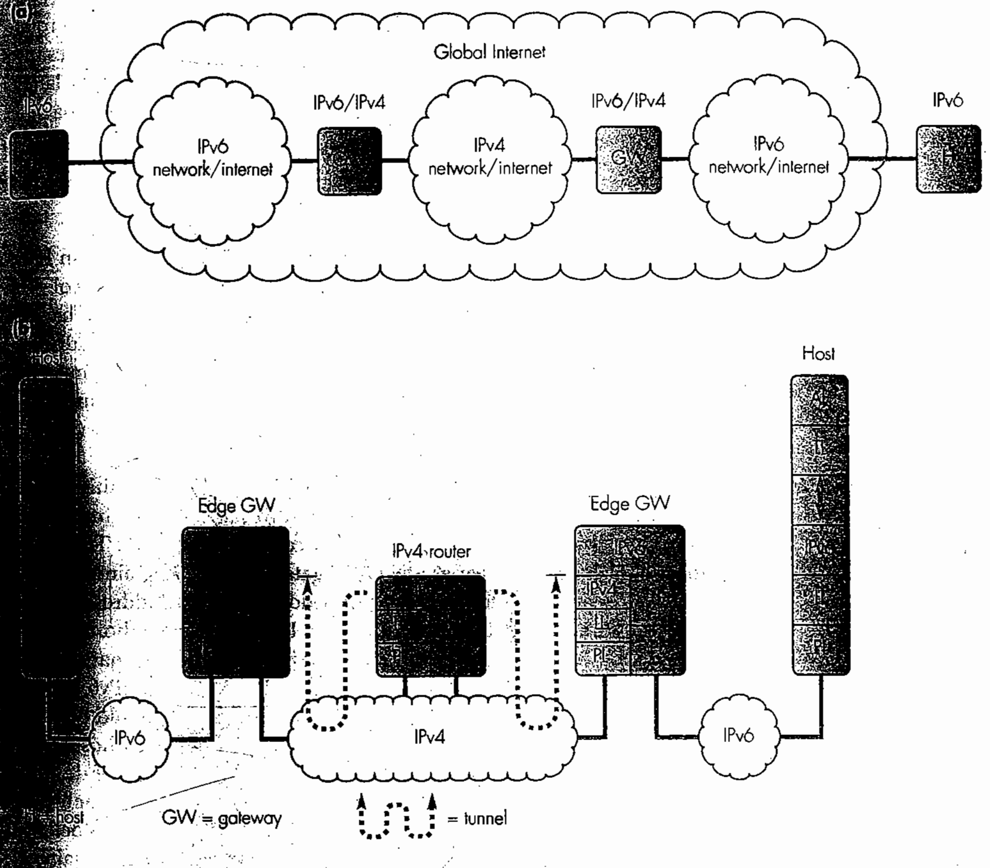

(a)
Global Internet
IPv6
IPv6 network/internet
IPv6/IPv4
IPv4 network/internet
IPv6/IPv4
GW
IPv6 network/internet
IPv6
(b)
Host
Edge GW
IPv4 router
Edge GW
IPv6
IPv4
IPv6
Host
IPv6
IPv4
IPv6
host
GW = gateway
= tunnel

Figure 9.37 IPv6/IPv4 interoperability using dual stacks and tunneling: (a) schematic; (b) protocols.

within an IPv4 packet. As we show in the figure, in order to do this, the edge gateways contain dual stacks each of which has a related IPv6/IP address associated with it. Normally, entered by network management, t routing table entry for the remote destination IPv6 host is the IPv4 address the remote edge gateway.

The IPv6 in each gateway, on determining from its routing table that (IPv6) packet should be forwarded to a remote IPv6 network via an IP tunnel, passes the packet to the IPv4 protocol together with the IPv4 addr of the remote gateway. The IPv4 protocol first encapsulates the IPv6 packet an IPv4 datagram/packet with the IPv4 address of the remote gateway in t destination address field. It then uses this address to obtain the (IPv address of the next-hop router from its own (IPv4) routing table and pr ceeds to forward the packet over the IPv4 network/internetwork. On recei of the packet, the IPv4 in the remote gateway, on detecting from its own rou ing table that it is a tunneled packet, strips off the IPv4 header and passes th payload – containing the original IPv6 packet – to the IPv6 layer. The latt then forwards the packet to the destination host identified in the pack header in the normal way.

9.11.3 Translators

A third type of interoperability requirement is for a host attached to an IP network – and hence having an IPv6 address and using the IPv6 protocol – t communicate with a host that is attached to an IPv4 network – hence havi an IPv4 address and using the IPv4 protocol. In this case, both the address and the packet formats are different and so a translation operation must carried out by any intermediate routers/gateways. As we show in Figure 9. the translations can be performed either at the network layer – part (a) – the application layer – part (b).

Using the first approach, on receipt of an IPv6/IPv4 packet, this is co verted into a semantically equivalent IPv4 /IPv6 packet. This involv **network address translator** (NAT) and a **protocol translator** (PT). The in mediate gateway is then known as a **NAT-PT gateway**. As we explained ea in Section 9.10.2, an IPv4 address can be embedded in an IPv6 address. He to send a datagram/packet from a host with an IPv6 address to a host wit IPv4 address, the NAT in the gateway can readily obtain the destination I address from the destination address in the main header of the IPv6 pack The issue is what the source address in the IPv4 packet header should be.

A proposed solution is for the NAT to be allocated a block of hostids f the destination IPv4 network. These, together with the netid of the destin tion network, then form a block of unique IPv4 addresses. For each ne call/session, the NAT allocates an unused IPv4 address from this block f the duration of the call/session. It then makes an entry in a table containin the IPv6 address of the V6 host and its equivalent (temporary) IPv4 addres

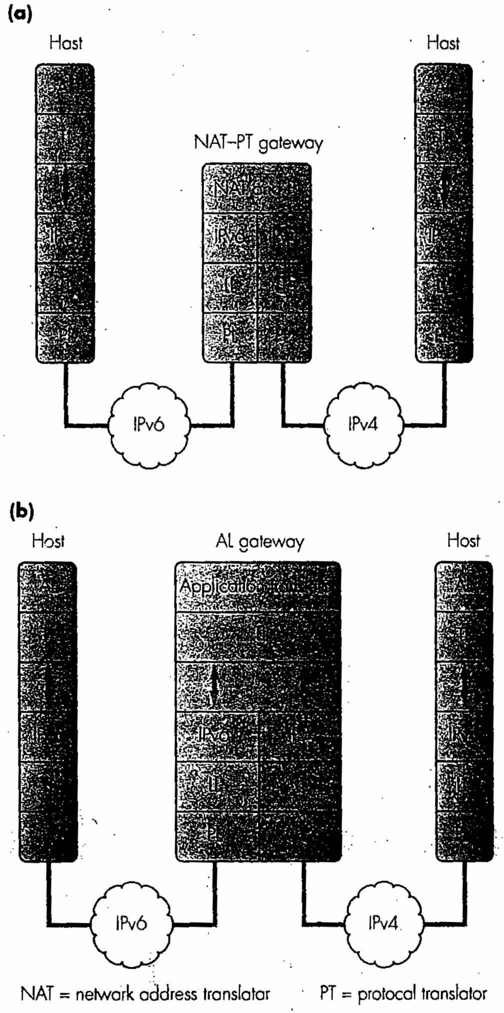


Figure 9.38 IPv6/IPv4 interoperability using translators: (a) network level; (b) application level.

The NAT translates between one address and the other as the packet is relayed. A timeout is applied to the use of such addresses and, if no packets are received within the timeout interval, the address is transferred back to the free address pool.

The protocol translation operation is concerned with translating th remaining fields in the packet header and, in the case of ICMP messag converting ICMPv4 messages into and from ICMPv6 messages. As we in cated in Section 9.10.1, most of the fields in the IPv6 main header have th same meaning as those in the IPv4 header and hence their translation is rel tively straightforward. In general, however, there is no attempt to transla the fields in the options part. Similarly, since ICMPv6 messages have differen *type* fields, the main translation performed is limited to changing this fiel For example, the two ICMPv4 query messages have *type* values of 8 and 0 and the corresponding ICMPv6 messages are 128 and 129.

The use of a NAT-PT gateway works providing the packet payload doe not contain any network addresses. Although this is the case for most applica tion protocols, a small number do. The FTP application protocol, fo example, often has IP addresses embedded within its protocol messages. In such cases, therefore, the translation operation must be carried out at the application layer. The associated gateway is then known as an **application level gateway** (**ALG**). This requires a separate translation program for each application protocol. Normally, therefore, most translations are performed at the network layer – using a NAT and a PT – and only the translations relating to application protocols such as FTP are carried out in the application layer.

Summary

Figure 9.39 opposite summarizes the various topics discussed in this chapter.

Exercises

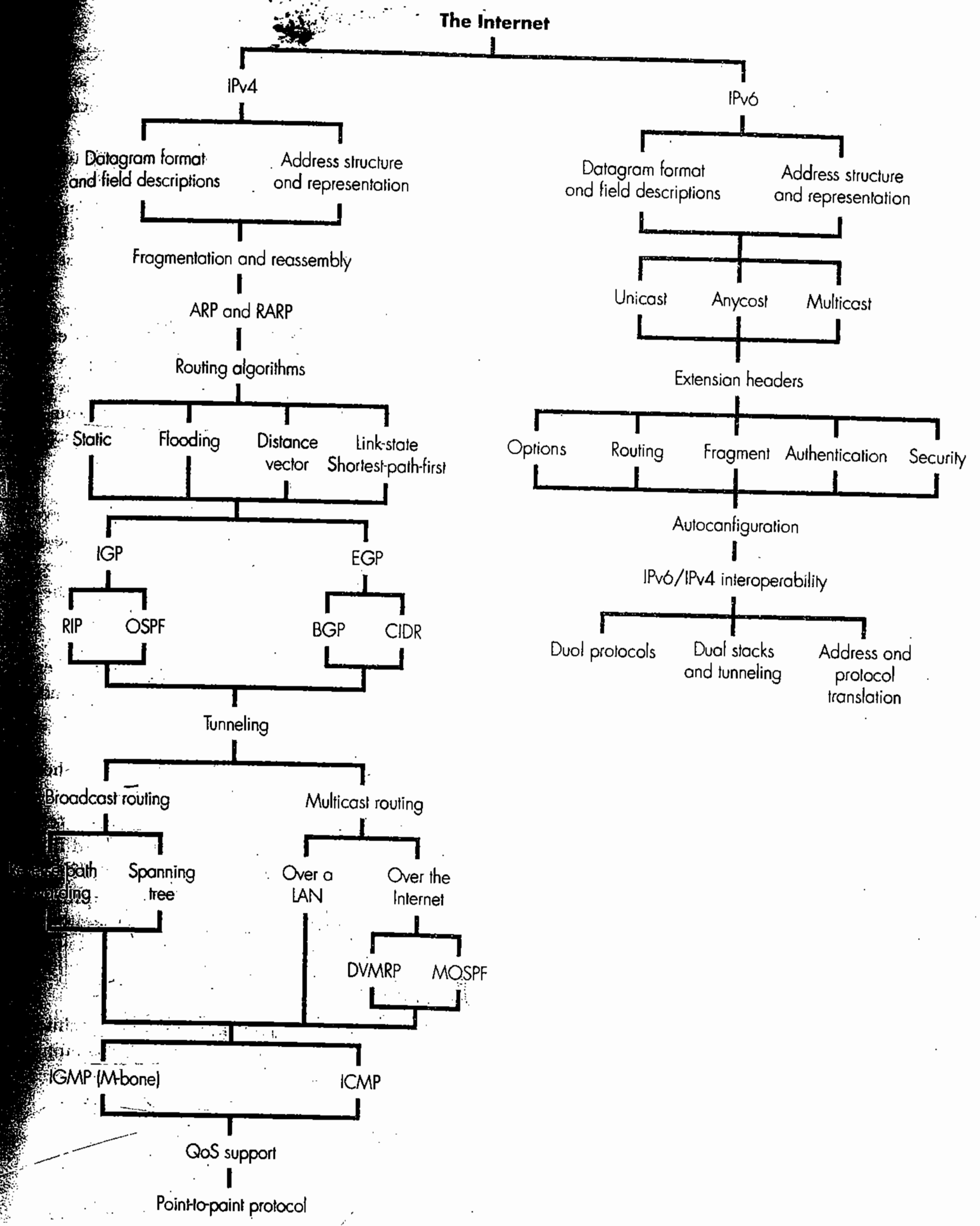
Section 9.1

9.1 With the aid of the network schematic shown in Figure 9.1, explain briefly the role of the following network components and protocols:
  (i) access network gateway,
  (ii) routing gateway,
  (iii) internet protocol (IP),
  (iv) datagram/packet.

9.2 With the aid of the protocol stack shown in Figure 9.2, explain briefly the meaning of the term "adjunct protocol" and how the IP in the destination host determines to which transport protocol – TCP/UDP – the contents/payload of a received IP datagram should be passed.

Section 9.2

9.3 In relation to the IP datagram/packet format shown in Figure 9.3, explain the role of the following header fields:
  (i) IHL,
  (ii) TOS,
  (iii) Total length and Identification,
  (iv) flag bits,
  (v) Fragment offset,
  (vi) Time-to-live,
  (vii) Protocol,
  (viii Header checksum,
  (ix) Options.



The Internet
IPv4
IPv6
Datagram format and field descriptions
Address structure and representation
Fragmentation and reassembly
ARP and RARP
Routing algorithms
Static
Flooding
Distance vector
Link-state Shortest-path-first
IGP
EGP
RIP
OSPF
BGP
CIDR
Tunneling
Broadcast routing
Multicast routing
path
Spanning tree
Over a LAN
Over the Internet
DVMRP
MOSPF
IGMP (M-bone)
ICMP
QoS support
Point-to-point protocol
Datagram format and field descriptions
Address structure and representation
Unicast
Anycast
Multicast
Extension headers
Options
Routing
Fragment
Authentication
Security
Autoconfiguration
IPv6/IPv4 interoperability
Dual protocols
Dual stacks and tunneling
Address and protocol translation

Figure 9.39 Summary of the topics discussed in relation to the Internet.

ction 9.3

Assume a message block of 7000 bytes is to be transferred from one host to another as shown in the example in Figure 9.4. In this instance, however, assume the token ring LAN has an MTU of 3000 bytes. Compute the header fields in each IP packet shown in the figure as it flows
- (i) over the token ring LAN,
- (ii) over the Ethernet LAN.

State why fragmentation is avoided whenever possible and the steps followed within each host IP to achieve this.

ction 9.4

Explain the meaning of the term "IP address class" and why these classes were created. Hence, with the aid of the three (unicast) address classes identified in Figure 9.5, identify a particular application for each class.

7 State the meaning of the following addresses:
- (i) an address with a netid of all 0s
- (ii) an address with a netid of all 1s
- (iii) an address with a hostid of all 0s
- (iv) an address of all 1s.

8 Explain the meaning of the term "dotted decimal". Hence derive the netid and hostid for the following IP addresses expressed in dotted decimal notation:
- (i) 13.0.0.15,
- (ii) 132.128.0.148,
- (iii) 220.0.0.0,
- (iv) 128.0.0.0,
- (v) 224.0.0.1.

.9 With the aid of an example, explain why subnetting was introduced. Hence state the meaning of a subnet router and an address mask.

.10 A site with a netid of 127.0 uses 20 subnet routers. Suggest a suitable address mask for the site which allows for a degree of expansion in the future. Give an example of a host IP address at this site.

Section 9.5

9.11 Define the terms "IP address", "MAC ad and "hardware/physical address". Also the terms "address-pair" and "ARP cache

9.12 In relation to the simple network top shown in Figure 9.7, explain why:
- (i) on receipt of an ARP request me each host retains a copy of the IP/ address-pair of the source host ARP cache
- (ii) on receipt of an ARP reply message ARP in the source host makes an en the IP/MAC address-pair in its own ca
- (iii) the LAN port of the gateway keeps a of the IP/MAC address-pair from ARP request and reply message th receives.

9.13 Explain the role of a proxy ARP. Hence exp how an IP packet sent by a host at one si routed to a host at a different site. Also exp how the reply packet is returned to the that sent the first packet.

9.14 Explain how the reverse ARP is used to en a diskless host to determine its own IP add from its local server.

9.15 With the aid of the two frame formats show Figure 9.8, explain:
- (i) how the MAC sublayer in the rece determines whether a received fra in the Ethernet format or IEEE802.
- (ii) the number of pad bytes required each frame type.

Section 9.6

9.16 With the aid of the global Internet archite shown in Figure 9.9, explain the meaning of the following:
- (i) continental backbone network,
- (ii) core backbone network,
- (iii) regional/national network,
- (iv) access network gateway,
- (v) multiprotocol gateway.

9.17 State the meaning of the following terms relating to the routing of packets over the Internet:
- (i) line cost,
- (ii) path cost,
- (iii) hopcount,
- (iv) routing metric,
- (v) shortest path.

9.18 With the aid of the routing table entries shown in Figure 9.11, explain the meaning of the terms:
- (i) static routing tables,
- (ii) next-hop routing,
- (iii) optionality principle,
- (iv) alternative paths.

9.19 With the aid of the broadcast diagram shown in Figure 9.12, explain:
- (i) why the broadcast following the route via R2 is assumed to arrive first
- (ii) how duplicate copies of a packet are determined by R3
- (iii) how the number of copies of the packet produced is limited
- (iv) why flooding is an example of an adaptive/dynamic routing algorithm.

9.20 In relation to the distance vector algorithm, with the aid of the example shown in Figure 9.13, explain:
- (i) the meaning of the term "connictivity/adjacency table" and how the table's contents are obtained
- (ii) how the final routing table entries for R3 are built up
- (iii) how a packet from a host attached to netid3 is routed to a host attached to netid1
- (iv) the limitations of the algorithm including how looping may arise.

9.21 Assuming the connectivity/adjacency tables given in Figure 9.14(a), show how the overall network topology is built up by router R3 using the link state algorithm. Hence derive the contents of netid location table for R3.

9.22 Assuming the initial network topology shown in Figure 9.15(a), use the Dijkstra algorithm to derive the shortest paths from R3 to each other router. State the meaning of the terms "tentative" and "permanent" relating to the algorithm and the implications of alternative paths/routers.

9.23 Using the set of link state, routing, and connectivity tables for R1, R2, and R3, explain how a packet received by R3 from G3 with a destination netid of 1 is routed using hop-by-hop routing.

9.24 Explain how a packet received by R3 from G3 with a destination netid of 1 is routed using source routing. Include how the routing tables you use are derived.

9.25 In relation to the link-state algorithm, explain why each link-state message contains a sequence number and a timeout value. How are these used?

9.26 With the aid of the generalized Internet architecture shown in Figure 9.18, state a suitable routing algorithm for use
- (i) within a single bridged LAN,
- (ii) within a single site comprising multiple different LANs interconnected by subnet routers,
- (iii) within a single autonomous system such as a regional/national network,
- (iv) within the core network of the Internet.

9.27 In relation to the BGP, with the aid of the message types shown in Table 9.1, explain how the following functions are performed:
- (i) neighbor acquisition and termination,
- (ii) neighbor reachability,
- (iii) routing update.

9.28 State the reason why classless inter-domain routing (CIDR) was introduced.

9.29 Derive the range of class C addresses that are available. Hence, using the list of continental addresses given in section 9.6.6, explain how the routing of packets across the global backbone network is carried out.

: why the lack of a fixed division point in class C addresses with CIDR means a er/gateway is unable to route a packet. ce explain how this is overcome.

:xample of how when using CIDR a host :hed to one network can produce a match the mask of a second network was illus-:d in Example 9.5. How would this be come in practice?

lain the term "tunneling" and when it is l. Hence with the aid of the schematic dia-n shown in Figure 9.20, explain how the : on the lift of the diagram sends an IP data-n/packet to a host attached to the rnet. Include in your explanation the role ıe two multiprotocol routers.
tate an application of tunneling IP packets r an IP network.

at are the aims of both the reverse path for-ding algorithm and the spanning tree adcast algorithm?

: the final routing tables and broadcast uence relating to the reverse path forward- algorithm shown in Figure 9.21 to explain y only the (broadcast)* packet received by 5 from SR3 is broadcast at the fourth stage. ıat is the number of duplicate broadcasts t occur?

ıuming the network topology shown in ıure 9.22(a) and that SR3 is access gateway, termine the spanning tree derived by each ›net router. Use this to derive the broadcast ıuence.

relation to multicasting over a LAN, scribe how IANA controls the allocation of ılticast addresses. Also explain how the 48-MAC address and 28-bit IP address of a host : derived from the allocated address. Hence ıh the aid of the schematic diagram shown in ʒure 9.23(b), describe how a host joins a

multicast session that is taking place over th LAN. Include the role of the multicast addre table and group address table held by each member of the group.

9.37 What is the meaning of the term "multicas router"? Outline the sequence of steps that are followed to route an IP packet with a multica address over the Internet.

9.38 Assume the same topology, multicast addres table contents, routing table contents, and routing table entries as shown in Figure 9.24 Assuming the DVMRP, explain how a packe arriving from one of its local networks with a multicast address of C is routed by MR3 to al the other MRs that have an interest in thi packet.

9.39 Repeat Exercise 9.37 but this time using the MOSPF routing protocol and the spanning tree shown in Figure 9.25(b).

9.40 What is the role of the IGMP protocol?
With the aid of the example shown in Figure 9.26, explain how a host that is attached to a local network/subnet of an MR joins a multi-cast session. Include in your explanation the table entries retained by both the host and the MR and how multicast packets relating to the session are then routed to the host.

9.41 With the aid of the example shown in Figure 9.26, explain the procedure followed when a host that is attached to a local network/subnet of an MR leaves a multicast session.

9.42 The multicast backbone (m-bone) network shown in Figures 9.24 and 9.25 comprised a set of multicast routers interconnected by single links. In practice, these are logical links since each may comprise multiple intercon-nected routers that do not take part in multicast routing. Explain how a multicast packet is sent from one mrouter to another using IP tunneling.

Section 9.7

9.43 Explain briefly the role of the ICMP protocol and the different procedures associated with it. Hence explain how the path MTU discovery procedure is used to determine the MTU of a path/route prior to sending any datagrams.

Section 9.8

9.44 Discuss the reasons why improved levels of QoS support are now being used within the Internet.

9.45 Describe the role and principle of operation of the following control mechanisms used within Internet routers:
(i) token bucket filter,
(ii) weighted fair queuing,
(iii) random early detection.

9.46 Define the three different classes of service used with the Intserv scheme. With the aid of the network topology shown in Figure 9.28 (a) describe the operation of the resource reserva-ion protocol (RSVP). Include in your description the meaning/role of the following:
(i) path, reserve, and path-tear messages
(ii) path-state table,
(iii) cleanup timer,
(iv) soft-state.

9.47 Define the usage of the type of service (ToS) field in each packet header with the DiffServ scheme including the meaning of the term "DS packet codepoint".

9.48 With the aid of the general architecture shown in Figure 9.28(b), describe the operation of the DiffServ scheme. Include in your descrip-tion the meaning/role of the following components of an ingress router;
(i) behavior aggregate,
(ii) traffic meter module,
(iii) MF classifier,
(iv) marker module,
(v) shaper/dropper.

Also explain the meaning/role of the following components of a core router:
(i) BA classifier,
(ii) per-hop behavior,
(iii) expedited forwarding,
(iv) assured forwarding.

Section 9.9

9.49 Identify a selection of the applications of the point-to-point protocol (PPP) within the Internet. With the aid of the PPP frame format shown in Figure 9.29, explain the meaning/use of the following fields:
(i) opening and closing flags with zero bit insertion,
(ii) the byte stuffing rules used with asyn-chronous transmission lines.

9.50 State the role of the protocol ID field in the header of a PPP frame. Hence describe in out-line the features supported by the link control protocol (LCP) associated with PPP.

Section 9.10

9.51 Discuss the reasons behind the definition of IP version 6, IPv6/IPng, including the main new features associated with it.

9.52 With the aid of the frame format shown in Figure 9.30(a), explain the role of the follow-ing fields in the IPv6 packet header:
(i) traffic class,
(ii) flow label,
(iii) payload length (and how this differs from the total length in an IPv4 packet header),
(iv) next header,
(v) hop limit,
(vi) source and destination addresses.

9.53 In relation to to IPv6 addresses, with the aid of the prefix formats shown in Figures 9.31(a) and (b), explain the meaning/use of:
(i) address aggregation,

10

Broadband ATM networks

10.1 Introduction

As we explained in Section 1.3.5, broadband ISDN (B-ISDN) was designe from the outset to support multimedia communication applications involvi text, high-resolution images, speech, audio, and video, either singly o number integrated together in some way. At the time of the design (and st dardization) of B-ISDN, the prevailing compression algorithms associat with the different media types were such that the anticipated peak chann bandwidth per call would be well in excess of the 2 Mbps offered by a (na rowband) ISDN.

Since each call may involve different types of media, the transmission an switching system chosen for the network is independent of both the bit ra and whether the source information is in the form of blocks or a variable ra bitstream. The chosen method involves the source information associat with each call being first converted into small fixed-sized packets known a **cells**. And since the rate of generation of the source information varies, so th rate of entry of cells into (and through) the network may also vary. Henc instead of allocating a fixed portion of transmission bandwidth per call, th cell streams relating to different calls are multiplexed together on a statistica basis. The switching units then operate using a form of packet switching called **cell switching**.

The adoption of a small fixed-sized cell means that cell switches can operate at a much higher rate than variable-length packet switches, so cell switching is also known as **fast packet switching**. Also, because the cells relating to different calls have varying time intervals between them during both transmission and switching, this mode of operation is known as the **asynchronous transfer mode** (ATM) and networks that operate in this way (broadband) ATM networks.

Having adopted a small cell size and a packet-switching mode of operation, the next decision to be made was whether the network should operate in a connection-oriented (CO) or a connectionless (CL) mode. As we showed in Figure 1.22 and explained in the accompanying text, in a CL mode network, each packet requires the full networkwide address of both the source and destination end systems in its header. The adoption of a small cell size, however, meant that the header may then be disproportionately large compared with the actual cell contents. So in order to keep the header small, the CO mode of operation was chosen. In a packet-switched network, this involves a path/route through the network being established prior to the transfer of any information packets and, on completion of the call, the path being closed down. The resulting path is called a virtual circuit and, in order for each packet-switching exchange to relate incoming packets to a specific virtual circuit, a unique identifier is assigned to the call on each line making up the virtual circuit as this is set up. The identifier used on each line is called the virtual circuit identifier (VCI) and, by assigning a new VCI on each line, only sufficient identifiers are required to identify the packets relating to the different calls on each line rather than the different calls in the total network. This same mode of operation is used in ATM networks.

The final decision was the size of the payload of a cell. Small cells have advantages for constant bit rate traffic since only a short delay is experienced as successive bytes relating to the same call are assembled and disassembled into and from cells at the network interface. Conversely, since each cell must contain additional routing information, small cells have the disadvantage that the overheads associated with each cell (in terms of transmission and switching) are higher. A compromise was reached by the various international standards bodies and a cell size of 53 bytes/octets was chosen. This comprises a 48-byte payload (information) field with a 5-byte header for the VCI and other fields. No error control is performed on cells within the network and hence no sequence numbers are required for retransmission purposes. However, the header does contain error check bits to detect the presence of transmission errors in the various header fields.

Because of the predicted high bit rates per call, an optical fiber access network was to be used to connect subscriber premises to the core network. As we indicated earlier, however, the rapid advances that have taken place in compression mean that the bit rate required for each call is significantly less than originally thought. As a result, existing access networks such as associated with an ISDN and a PSTN can now support a range of multi communication applications. Hence the introduction of a B-ISDN acces work has been postponed by most telecom providers. Nevertheles asynchronous transfer mode of working has been adopted for a numb other networks in which high bandwidth is required. As we showed in Fi 1.5, examples include ATM LANs and MANs, the latter providing a swit high-speed LAN interconnection facility. Also, as the bit rate of the trans sion lines used in the Internet backbone network has increased – owin the increasing number of multimedia applications – the cell-switching m has been incorporated into the design of some of the high-speed routers u in the core backbone. In this chapter, we present further details of the op tion of cell-switching networks in Sections 10.2 to 10.4 and the operatio ATM LANs in Section 10.5. We then describe the operation of ATM MAN Section 10.6 and show how they are being used as the basis of broadb wide area networks in Section 10.7.

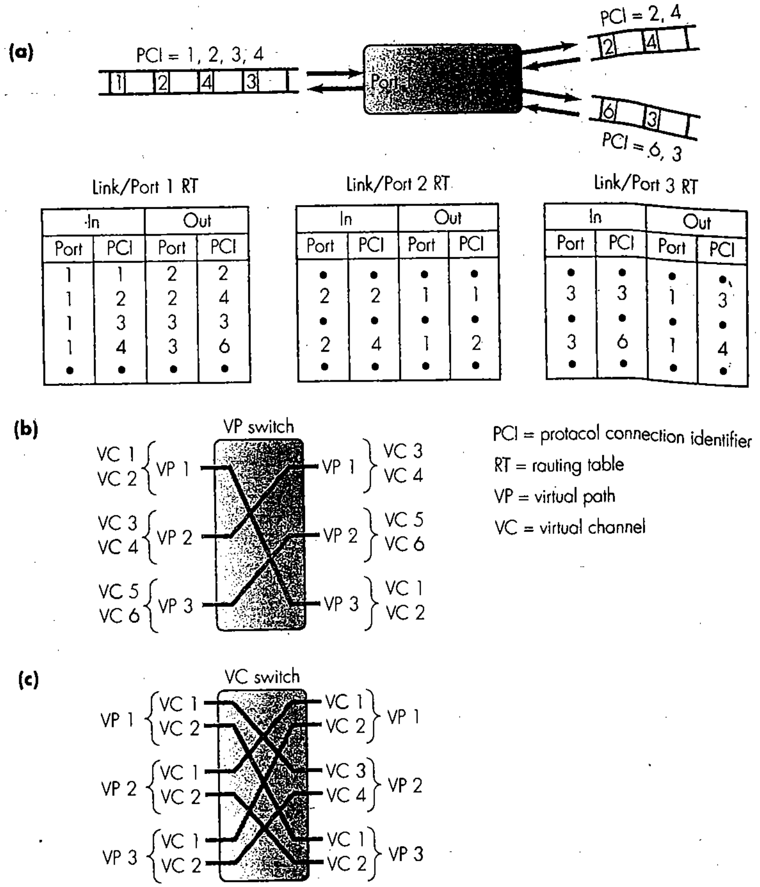
10.2 Cell format and switching principles

As we have explained, prior to any information cells being sent, a virtual c cuit is first established. In an ATM network, the virtual circuit identifier us on each link is known as the **protocol connection identifier (PCI)**. The pr ciple of the routing scheme used is shown in Figure 10.1(a).

Associated with each incoming link/port is a routing table that contai for each incoming PCI, the corresponding outgoing link/port and the n PCI to be used. The routing of cells in both directions along a route is th very fast as it involves a simple look-up operation. As a result, cells from ea link can be switched independently and at very high rates. This allows par lel switch architectures to be used and high-speed transmission lines in t gigabit range, each operating at its maximum rate.

In practice, the PCI is made up of two subfields: a **virtual path identi (VPI)** and a **virtual channel identifier (VCI)**. Routing can be perform using either one or a combination of the two. Two examples are shown Figure 10.1. In part (b), switching is performed on virtual paths and the V within each virtual path remain unchanged. In part (c) switching is pe formed on the virtual channels within each virtual path independently a the virtual paths simply terminate at each switch port.

An example of the use of virtual path switching is when multiple ca originating at the same network entry point are all intended for the same de tination exit point. Each individual call is assigned a separate VCI and th calls are multiplexed together at the source interface onto a single virtu path. The multiplexed set of calls are then switched using the VPI field onl and hence all follow the same path through the network. In this way, the se



Link/Port 1 RT

In		Out	
Port	PCI	Port	PCI
1	1	2	2
1	2	2	4
1	3	3	3
1	4	3	6
•	•	•	•

Link/Port 2 RT

In		Out	
Port	PCI	Port	PCI
•	•	•	•
2	2	1	1
•	•	•	•
2	4	1	2
•	•	•	•

Link/Port 3 RT

In		Out	
Port	PCI	Port	PCI
•	•	•	•
3	3	1	3
•	•	•	•
3	6	1	4
•	•	•	•

PCI = protocol connection identifier
RT = routing table
VP = virtual path
VC = virtual channel

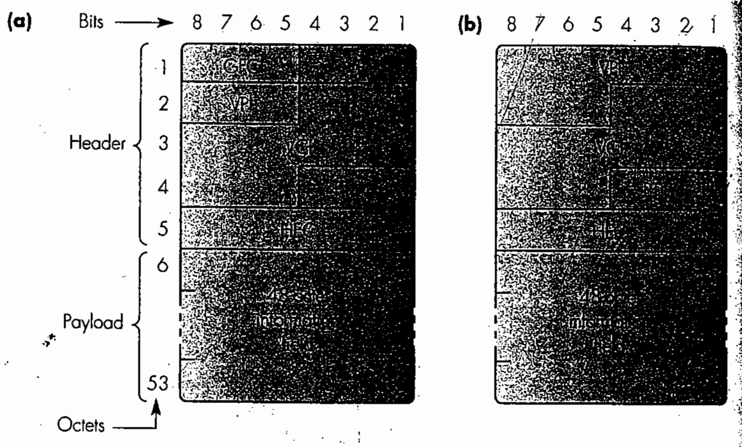
Figure 10.1 Cell switching principles: (a) routing schematic; (b) VP routing; (c) VC routing.

of VCIs remain unchanged and are used at the destination to identify (demultiplex) the individual calls.

An example of the use of virtual channel switching is when each call at the network entry point is intended for a different destination. In this case, each call is again assigned a different VCI but switching is also performed using the VCI field. The VPI field then has local significance to each link only and is used, for example, to allow calls to be multiplexed together for transmission purposes.

The format of each cell is shown in Figure 10.2 and, as we can see, the header is made up of six fields. Their functions are as follows:

- **generic flow control (GFC):** this is present only in cells transferred over the user–network interface (UNI) and is included to enable a local switch to regulate – flow control – the entry of cells by a user into the network. Within the network, cells transferred over the interexchange links – known as the network–network interface (NNI) – do not contain this field and the four bits are part of the VPI field;
- **virtual path identifier (VPI):** this is 8 bits at the UNI and, as just indicated, 12 bits at the NNI. As described previously, it is used for identification/routing purposes within the network;



PTI:

000 – user data, no congestion, SDU type 0
001 – user data, no congestion, SDU type 1
010 – user data, congestion, SDU type 0
011 – user data, congestion, SDU type 1
100 –
101 –
110 –
111 –
Network control

GFC = generic flow control
VPI = virtual path identifier
VCI = virtual channel identifier

PTI = payload type identifier
CLP = cell loss priority
HEC = header error checksum

Figure 10.2 ATM cell formats: (a) user–network segment; (b) within network, network–network interface.

- **virtual channel identifier (VCI):** a 16-bit field used for identification/routing purposes within the network;
- **payload type indicator (PTI):** indicates the type of information carried in the cell; the different types are shown in Figure 10.2. All cells containing user data have a zero in the most significant bit. The next bit indicates whether the cell has experienced excessive delay/congestion or not, and the third bit the service data unit (SDU) type – 0 or 1. We shall discuss its use in Section 10.4.1 in relation to the AAL5 service. The four remaining cell types are used for network control purposes;
- **cell loss priority (CLP):** within the network, the statistical multiplexing of cells on each link may occasionally result in cells having to be discarded during heavy load conditions. This field has been included to enable the user to specify a preference as to which cells should be discarded; CLP = 0 high priority, CLP = 1 low priority and hence discard first;
- **header error checksum (HEC):** generated by the physical layer and is an 8-bit CRC on the first 4-bytes of the header.

10.3 Switch architectures

The general structure of an ATM switch is shown in Figure 10.3(a). Each input link is terminated by an **input controller (IC)** which performs the routing of cells arriving at each link (port) to their required output link. This involves a simple look-up and mapping operation of the VPI/VCI in the header of the incoming cell into the corresponding output VPI/VCI. Normally, the output port number obtained from the routing table is used to determine the path to be followed through the switching fabric to the required output controller.

Because there is no reservation of slots on the output links, cells may arrive simultaneously at two (or more) input ports that require the same output port/link. This is handled in one of two ways: either the input controllers contain a set of cell buffers that hold the additional cell(s) or the buffering is provided in the **output controllers**. In both cases the buffers are organized in the form of a FIFO queue to ensure the cells from each input controller are output in the same order as they arrived. The output controllers simply forward received cells at the appropriate link bit (and hence cell) rate.

The main role of the **control processor** is to download routing information into the routing tables in each input controller. Normally, the routing information is received through the network either from a network management station or, if switched VCs are being used, from a signaling control point processor. In both cases, semipermanent VCs are used to relay the cell containing the routing information. Hence on arrival at the switch, the related cells are routed through the switching fabric directly from the input controller that receives the cells to the control processor. In addition, the

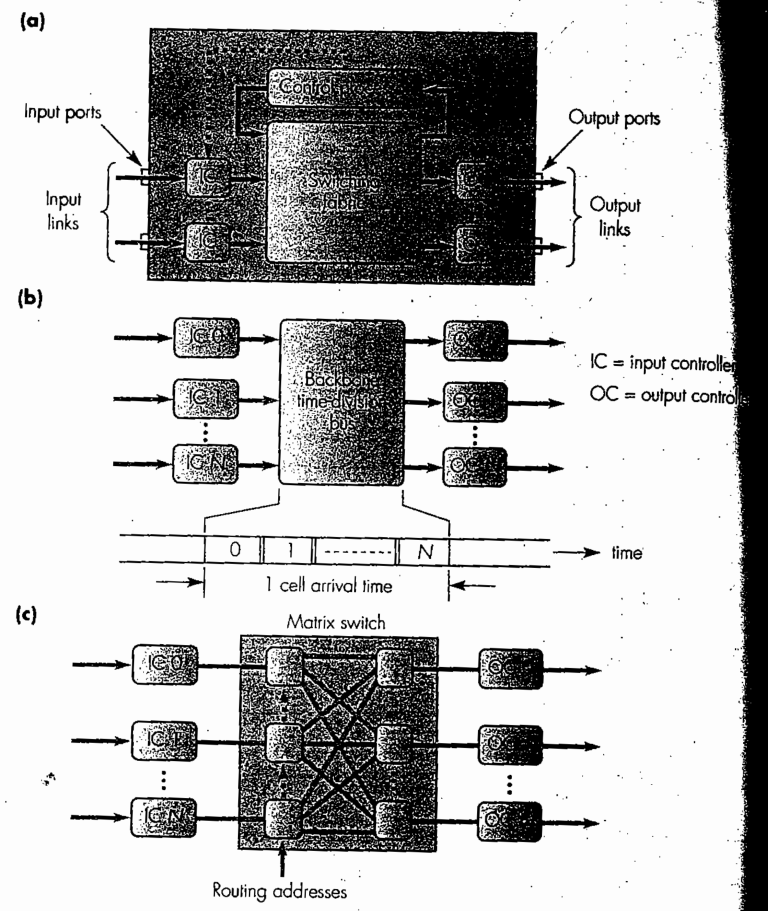


Figure 10.3 ATM switch architectures: (a) general structure; (b) tim division bus schematic; (c) fully-connected matrix switch.

control processor may generate network management messages itself – fau reports, performance statistics, and so on – and these are also routed throug the switch fabric to the required output controller for onward transmission t the network management location again using a semipermanent VC.

A number of alternative switch fabrics are used in ATM switches. They ca be classified as either time division or space division. Normally, all the inpu controllers are synchronized so that each set of incoming cells from all controllers is presented to the switch fabric in synchronism. The switch fabric also

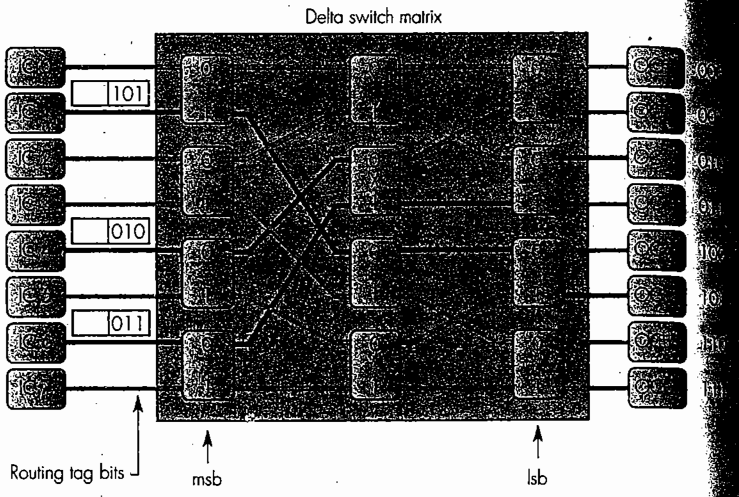
operates synchronously which means that the cells from each input controller are transferred to their required output controller in a single cell time.

A schematic diagram of a **time-division switch** is shown in Figure 10.3(b). In such switches we use a time-division backplane bus that is capable of transferring *N* cells – where *N* is the number of input ports – in a single cell arrival time. Each input controller is assigned its own cell (slot) time to transfer a cell over the backplane bus. The input controller appends the required output port number to the head of the cell and this is used by the set of output controllers to determine which output controller should read and buffer the cell. If more than one cell is received by an output controller in a single cell arrival time then these are queued in the controller. Also, for one-to-many communications (multicast), more than one output controller may be specified to receive the cell. Typically, this type of switch fabric is used in switch designs which have a relatively small number of ports, the number being limited by the speed of operation of the backplane bus and also the output controllers; for example, a 2.5 Gbps bus can support 16 × 155 Mbps or 4 × 622 Mbps duplex links/ports.

In a **space-division switch** the switch fabric comprises a matrix of interconnected **switching elements** that collectively provide a number of alternative paths through the switch. An example is shown in Figure 10.3(c). This is known as a **fully-connected switch matrix** since a path is provided from all input ports/controllers to all output ports/controllers. In the example shown, each input switching element is capable of passing a copy of each received cell to any of the output switching elements. The latter then receive the cells offered and pass them on to the output controller(s) for transmission. Although queuing is necessary in the input or output controllers, normally the aim is to avoid additional queuing within the switching fabric itself. To avoid internal queuing with this type of switch the cell transfer operation must be performed *N* times faster than the cell arrival rate, where *N* is the number of input ports. First the cell from the first input switching element is transferred, then the cell from the second element, and so on. Providing this can be done, no additional buffering is required within the switching matrix and the switch is said to be internally **non-blocking**.

We can deduce from Figure 10.3(c)that in a fully-connected switch the number of interconnection paths required through the switch – and hence output/input circuits associated with each switching element – grows as a function of N^2 and the speed of operation of the output switching element by *N*. In practice, this limits the maximum size of such switches. Hence most practical matrix switch designs use multiple switching stages, each made up of a number of smaller switching elements interconnected in a regular matrix. This also simplifies considerably the implementation of the switch fabric in integrated circuit form.

A switching fabric that comprises multiple switching stages is the **delta switch matrix**, an example of which is shown in Figure 10.4. As we can see, these switches are made up from an interconnected set of identical switching



X = blocking, cell with tag 011 assumed dropped

Figure 10.4 Delta switch matrix example.

elements. In this example, a 2 × 2 switching element is used although larg sizes are also used. In general, the number of switching elements per stage, is determined from the expression $X = M/N$ where M is the total number input lines and N is the number of inputs per switching element. Also, th number of stages required, Y, is determined from the expression $N^Y = M$. the example, $N = 2$ and $M = 8$ and hence three stages are required each co prising four switching elements.

The internal interconnections between switching elements are such th there is a path through the switch from any input to any output. Associat with each switching element is a routing control bit called the **routing tag** the tag bit is a binary 0, then a cell arriving on either input is routed to th upper output, while if the tag is a binary 1, then it is routed to the low output. As we can deduce from Figure 10.4, the same set of three routing ta bits will route a cell through the matrix from any input port to the sam output port. Such networks are said to be **self-routing**.

With this type of switch, to route cells through the switch matrix ea input controller simply reads the new PCI and output port number – th routing tag – from its routing table, writes the PCI into the cell header, an then appends the routing tag to the head of the cell. The switching eleme at each stage along the path through the matrix then uses its own bit fro the routing tag – most significant bit first – to perform its routing operatio In this way, routing is very fast and each cell arrives at its intended destinatio port regardless of the switch port on which it arrived.

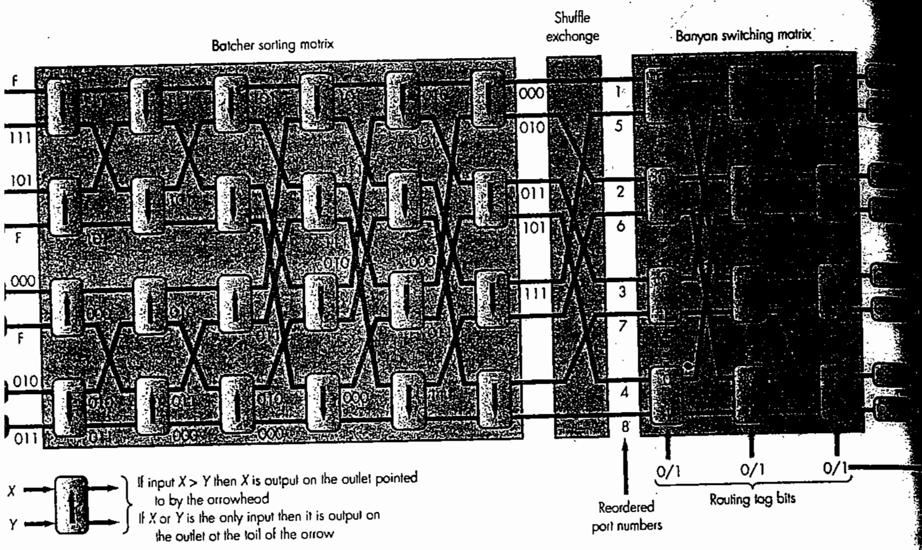
The disadvantage of this type of switching fabric is **blocking**. Three example paths through the switch are shown as bold lines in Figure 10.4. As we can see, although the cell addressed to port 5 (101) has an unimpeded path through the matrix, those addressed to ports 2 (010) and 3 (011) both arrive at the second switching element simultaneously. Both require the same output line and blocking is said to occur.

There are a number of ways of overcoming blocking. One approach is for the switching element to discard one of the two cells. It is assumed that this approach is used in the example and that the cell with a tag of 011 is dropped. In support of this, note that not all ports will enter a cell into the matrix during each cell arrival time since only in the heaviest load conditions do cells arrive contiguously at all inputs. This means that under normal loads, several ports will receive idle/empty cells which do not need routing and hence in practice the probability of blocking is small. Nevertheless, in general, discarding cells in this way can lead to an unacceptably high cell loss rate in large switches.

A second approach is to perform the switching operation several times faster than the cell arrival rate, for example by allowing each cell into the matrix after the preceding cell leaves the first stage switching element. There is a limit to the speed of operation of the switching elements and their interconnecting links and hence for larger switching fabrics this becomes impractical on its own. A third approach is to introduce buffering into each switching element but this has the disadvantage of introducing additional delay to the switching operation. In practice, a combination of all three approaches is used in practical switch designs.

An example of a switching fabric that avoids internal blocking is shown in Figure 10.5. It is known as the **Batcher–Banyan switch**. With a delta switch, blocking occurs when either different inputs all require a path through the switch to the same output or, the paths through the switch between different input and output ports involve a common output line from a switching element. In the Batcher–Banyan switch, blocking is avoided firstly, by ensuring that no two cells entering the switching matrix require the same output port and secondly, within the switching matrix itself, by ensuring that there are no common interconnecting links within the paths through the switch.

To satisfy the first condition, the buffering of cells is performed in the input controllers instead of the output controllers. Then, if two (or more) cells arrive simultaneously at different input ports that require the same output port, just one cell is selected for transfer across the switching fabric and the other is queued in the input controller until the next cell transfer time. To satisfy the second condition, as Figure 10.5 shows, the (Banyan) switching matrix is preceded by a (Batcher) **sorting matrix** and the two are interconnected using what is called a **shuffle exchange**. The combined effect is that all cells arriving at the switching matrix are ordered such that they each follow a unique path through the switching matrix. Hence, providing independent routing paths are available within each switching element, the


Batcher sorting matrix
Shuffle exchange
Banyan switching matrix
F
111
101
F
000
F
010
011
000
010
011
101
111
1
5
2
6
3
7
4
8
0/1
0/1
0/1
Reordered port numbers
Routing tag bits
X
Y
If input X > Y then X is output on the outlet pointed to by the arrowhead
If X or Y is the only input then it is output on the outlet at the tail of the arrow

ure 10.5 Batcher–Banyan switch matrix.

cells from all input ports can be switched simultaneously. The size o Batcher sorting network grows by $N (\log N^2)$ and can be considerable f large switches.

Another approach used in practical switch designs to reduce the blocki probability is a switching fabric that has multiple paths between each pair input and output ports. For example, with the simple delta switch, this can achieved by replicating the total switch matrix an appropriate number times. Each input controller uses a different matrix to transfer each c Alternatively, the basic Banyan switching matrix can be extended by add extra switching stages. In both cases, the number of additional stages can selected so that the blocking – and hence cell loss – probability is at acceptable level.

Another issue which has an impact on switch design is multicasti Recall from Section 9.6.9 that this requires all the cells from each workstatio in a (multicast) group to be sent to all the other workstations in the group. I practice, the most efficient way of achieving this is to use the switches to rou copies of cells to multiple destinations. Although this can be done readil with both the time-division and fully-connected switch designs, with matrix switches, because of their self-routing property, additional switching stages are needed if multicasting is to be supported. Further details relating to both these issues can be found in the bibliography for this chapter.

10.4 Protocol architecture

As we show in Figure 10.6, the ATM protocol architecture supports three separate application functions (planes). These are the control (C) plane, the user (U) plane and the management (M) plane. The protocols associated with the **C-plane** are concerned with signaling; that is, the setting up and clearing of on-demand VCs. Typically, these are set up and cleared using a signaling protocol set in the station communicating with a similar protocol set in the network **signaling control point**. The protocols in the **U-plane** depend on the application and, in general, communicate on a user-to-user (peer) basis with a similar protocol set in the destination station. The protocols in the **M-plane** are concerned with the management of the station; examples include reporting any error conditions that may arise during normal operation to the network management station and receiving notifications of the

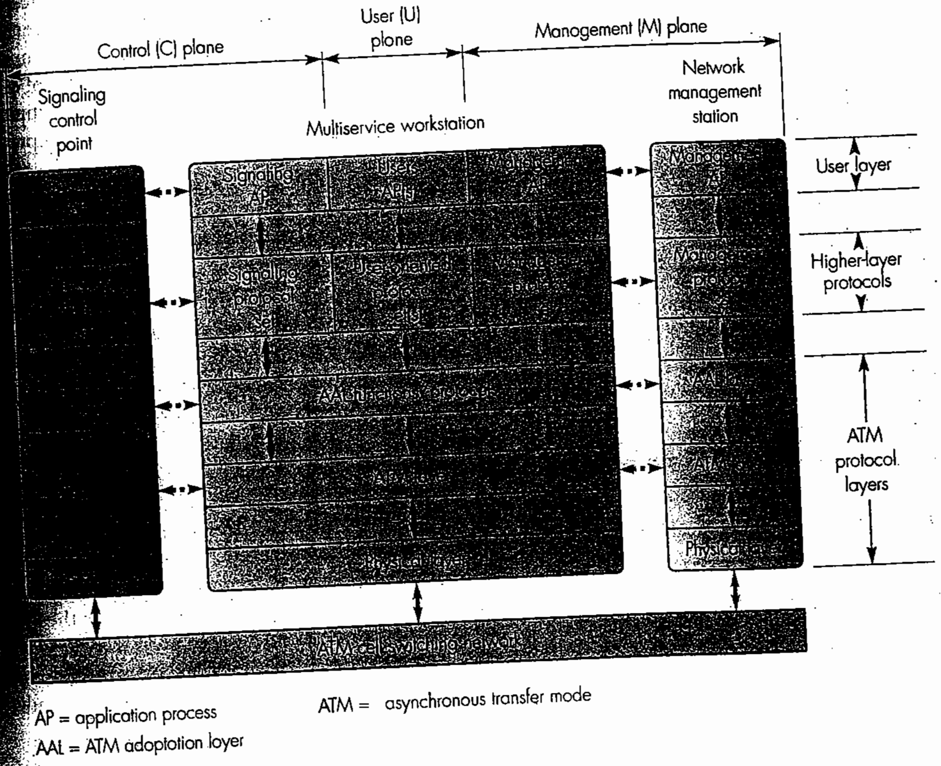

Control (C) plane
User (U) plane
Management (M) plane
Signaling control point
Multiservice workstation
Network management station
Signalling AP
User AP
Management AP
Signalling protocols
User-related protocols
AAL
ATM
Physical
User layer
Higher-layer protocols
ATM protocol layers
ATM cell switching network
AP = application process
AAL = ATM adaptation layer
ATM = asynchronous transfer mode

Figure 10.6 ATM protocol architecture.

virtual path/channel identifiers that have been allocated for permanent VCs (PVCs). These three application functions then use the services provided by the three lower ATM protocol layers for the transfer of the associated messages over the cell-based ATM switching network.

The ATM network supports a range of different services. The use of cell switching and transmission within the network is transparent to the upper application protocols which view the ATM network simply as a flexible facility for the transfer of information relating to any media type. To achieve this transparency, the highest of the three ATM layers is known as the **ATM adaptation layer** (AAL). As the name implies, it performs an adaptation (convergence) function between the service provided to the user layer above – for example the transfer of a data frame between two legacy LANs – and the cell-based service provided by the underlying ATM layer.

To support the various information sources, the AAL layer provides a range of alternative service types known as **service classes**. Associated with each service class is a different adaptation function/protocol which converts the source information into streams of 48-octet segments. It passes these to the **ATM layer** for transfer across the network. The ATM layer is concerned with adding the correct cell header information to each segment and multiplexing the cells relating to different connections into a single stream of cells for transmission over the network. It is also concerned with the demultiplexing of received cell streams and relaying their contents to the appropriate AAL protocol at the destination.

The **physical layer** can take on a number of different forms and depends on the type of transmission circuits being used. The upper part of the physical layer is known as the **transmission-convergence sublayer** and is concerned with such functions as the generation of the header check sequence in the cell header and the delineation of the cell boundaries. The lower part takes on different forms and is known as the **medium-dependent sublayer**. It is concerned with such functions as line coding and bit/clock synchronization. We shall discuss the operation of the AAL and ATM layer in more detail in the following two subsections.

10.4.1 ATM adaptation layer

The AAL provides a range of alternative service types/classes for the transport of the byte streams/message units generated by the various higher protocol layers associated with the U-, C-, and M-planes. It converts the submitted information into streams of 48-octet segments and transports these in the payload field of multiple ATM cells. Similarly, on receipt of the stream of cells relating to the same call, it converts the 48-octet information field contained within each cell into the required form for delivery to the particular higher protocol layer.

The service types are classified according to three criteria: the existence of a time relationship between the source and destination users (for example voice), the bit rate associated with the transfer (constant or variable), and the connection mode (connection-oriented or connectionless). Currently, five service types have been defined. They are referred to as AAL 1–5 and their interrelationship, based on these criteria, is illustrated in Figure 10.7(a).

Both AAL 1 (Class A) and AAL 2 (Class B) are connection-oriented and there exists a timing relationship between the source and destination users. The difference between the two is that AAL 1 provides a **constant bit rate (CBR) service** while AAL 2 provides a **variable bit rate (VBR) service**. An example use of AAL 1 is for the transfer of the constant bit rate byte stream associated with a voice call, for example, 1 byte per 125 µs. AAL 1 is also known as **circuit (switched) emulation**. An example use of AAL 2 is for the transmission of the variable bit rate stream associated with compressed video. Although video produces frames at a constant rate, a video codec will produce frames containing a variable amount of compressed data.

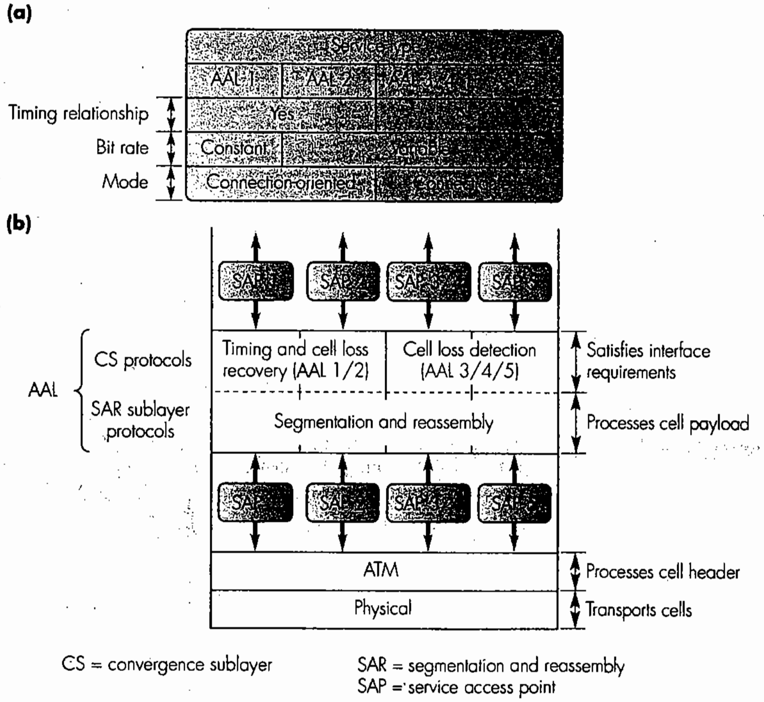


Figure 10.7 ATM adaption layer: (a) service class relationship; (b) sublayer protocols and their functions.

There is no timing relationship between source and destination w either AAL 3/4 (Class C/D) or AAL 5. Initially, AAL 3 was defined to prov a connection-oriented, VBR data service. Later, this service type was dropp and it is now merged with AAL 4. Both AAL 3/4 and AAL 5 provide a simi connectionless VBR service. An example use is for the transfer of data fram between two legacy LANs – through a remote bridge or router for exampl or for the transfer of frames containing multimedia information between workstation and a server. It is also used for the transfer of the message un associated with the signaling and management protocol sets in such wor stations. In all of these cases, as we shall see in Section 10.5.1, although th service offered is connectionless, the resulting cell streams produced by th AAL layer are transferred over previously established PVCs, for example, t the signaling control point or network management station.

In order to implement this range of services, the AAL comprises two sub layers as we show in Figure 10.7(b). The **convergence sublayer (CS)** perform a convergence function between the service offered at the layer interface an that provided by the underlying ATM layer. The **segmentation and reassembl (SAR)** sublayer performs the necessary segmentation of the source informa tion ready for transfer in the 48-octet payload field of a cell and also, the corresponding reassembly function at the destination prior to the delivery o the source information.

Since the submitted information differs for each service type, there is a different convergence function – and hence CS protocol – associated with each service type. Associated with each protocol is a **service access point (SAP)** which is used to direct all information submitted for transfer – the ser vice data unit (SDU) – to the appropriate CS protocol. Similarly, there are four different types of SAR protocol, each with its own PDU structure. As we shall now see, each utilizes the 48-octet information field in each cell in a different way.

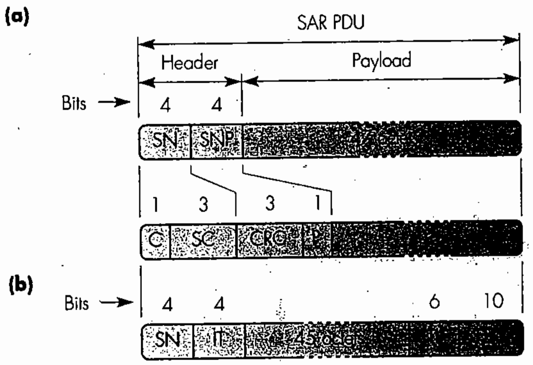
AAL 1

For this type of service the CS protocol endeavors to maintain a constant bi rate stream between the source and destination SAPs. The bit rate can range from a few kilobits per second -- for example, for a single compressed voice call – to tens of megabits per second – for example, for a compressed video However, the agreed rate must be maintained, even when occasional cel losses or cell transfer time variations occur. Cell losses are overcome in an agreed way, for example, by inserting dummy bits/bytes into the delivered stream. Cell transfer delay variations are compensated for by buffering segments at the destination: the output of the bits/bytes relating to a call is started only after a predefined number of segments have been received, this number being determined by the user bit rate. Typical figures are two segments at kilobit rates and 100 segments at megabit rates. The use of buffering at the destination also provides a crude way of overcoming any small variations between the input rate at the source interface and the output rate at the destination; for example, if each is based on a separate clock.

The format of each PDU associated with the SAR protocol is shown in Figure 10.8(a). To detect segment losses, the first octet contains a 4-bit *sequence number* (*SN*) and an associated 4-bit *SN protection* (*SNP*) field which is used to protect the sequence number against single bit errors. The sequence number is itself made up of a 3-bit *sequence count* field – for the detection of lost cells – and a single *convergence sublayer indication bit*. The latter can be used for the transfer of timing and/or other information relating to the payload field. The sequence number protection field comprises a 3-bit CRC, generated by the polynomial $x^3 + x + 1$, and an even parity bit. The latter is used to detect errors in the CRC.

AAL 2

With this type of service, although there is a timing relationship between the source and destination SAPs – determined by the frame rate, for example, for compressed video – the amount of information associated with each compressed frame may vary from one frame to the next. The CS protocol at the source receives bursts of information at the frame rate with each burst containing a variable amount of information. Hence the peer CS protocol at the destination must endeavor to output the received information in this same way even when occasional cell losses or cell transfer time variations occur. With AAL 2, the time variations are overcome using similar techniques to those described for AAL 1.



SN = sequence number
SNP = sequence number protection
C = convergence sublayer indication
SC = sequence count
P = even parity bit
IT = information type
LI = length indicator
FEC = forward error correction

Figure 10.8 SAR protocol data unit types: (a) AAL 1; (b) AAL 2.

The format of each PDU associated with the corresponding SAR protoco is shown in Figure 10.8(b). As with AAL 1, the sequence number is presen detect (and recover from) lost cells and to carry timing information. Th *information type* (*IT*) field indicates either the position of the segment in re tion to a submitted message unit – for example, a compressed frame – whether the segment contains timing or other information. The three seg ment types relating to positional information are beginning of messag (BOM), continuation of message (COM), and end of message (EOM). Also because of the variable size of submitted message units, the last (EOM) seg ment may not be full and hence the length indicator (LI) at the tail indicate the number of useful bytes in the segment. Finally, the forward error correc tion (FEC) field enables bit errors to be detected and a selection corrected

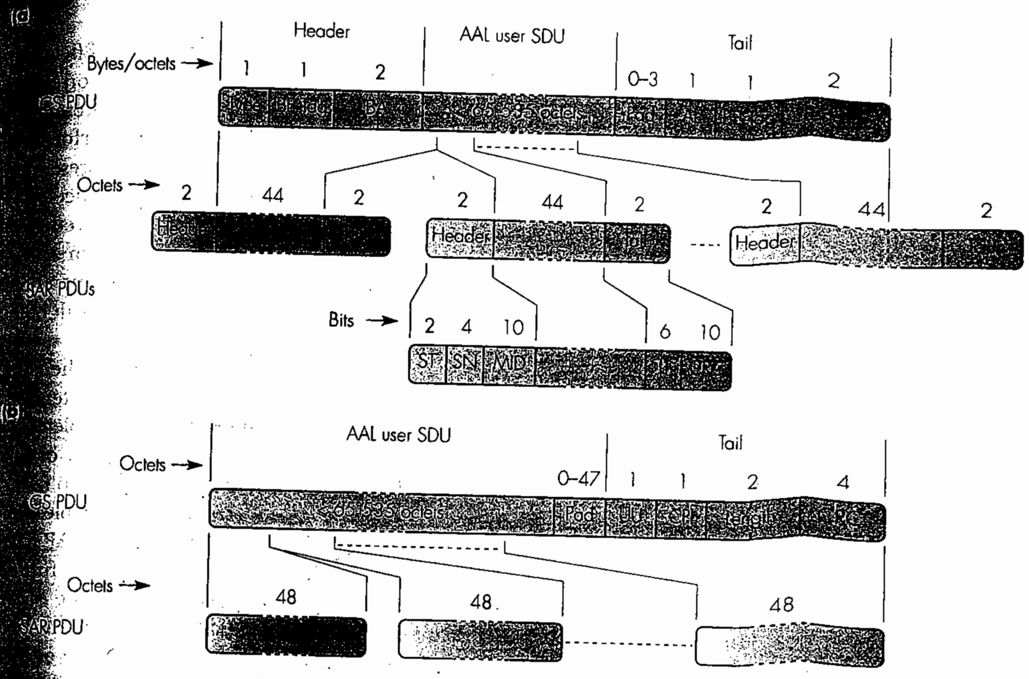
AAL 3/4

AAL 3 was defined initially to provide a connection-oriented data transfer se vice. Later, this type of service was dropped and combined with AAL 4. AA 3/4 provides a connectionless data transfer service for the transfer of variabl length frames up to 65 535 bytes in length. Error detection and other field are added to each frame prior to its transfer and the resulting frame i padded so that it is an integral multiple of 32 bits.

The operation of AAL 3/4 is best described by considering the format o the PDUs associated with both the CS and SAR protocols. The format of th additional fields added by the CS protocol to each submitted user SDU – a the corresponding SAP – is shown in Figure 10.9(a). The figure also show how the resulting CS PDU is segmented by the SAR protocol into multip 48-octet SAR-PDUs.

The header and trailer fields added to the submitted SDU by the CS pr tocol at the source are used by the peer CS protocol at the destination t detect any missing or malformed SDUs. The *PDU-type* field is a legacy of th earlier AAL 3 which required multiple types of PDU. It is set to zero with AA 3/4. The *begin-end* (*BE*) *tag* is a modulo-256 sequence number and is repeate in the trailer for added resilience. It enables SDUs to be delivered at the us interface in the same sequence as they were submitted although, again, th facility is not normally used with a connectionless service. The *buffer allocatio* (*BA*) field is inserted in the header by the source to help the CS protocol a the destination allocate an appropriate amount of (buffer) memory for th complete SDU. At the trailer, the *pad* field is used to make the total number o octets in the complete CS-PDU an integral multiple of 4 octets. Similarly, th *alignment* (*AL*) field is a single (dummy) octet to make the trailer 4 octets als Collectively this leads to easier memory management at the destination. Th *length* field indicates the total length of the complete PDU and this is used b the receiving protocol to detect any malformed SDUs.

On receipt of each CS-PDU, the SAR protocol segments this into multipl 48 octet segments – SAR-PDUs – as shown in Figure 10.9(a). In the header the *segment type* (*ST*) indicates whether the segment is the first, continuation last, or only segment resulting from the segmentation of the CS-PDU. The



CS = convergence sublayer
SAR = segmentation and reassembly
BE = begin–end
BA = buffer allocation
AL = alignment
ST = segment type
SN = sequence number
MID = message identifier
LI = length indicator
UU = AAL layer user-to-user identifier
CPI = common part identifier

Figure 10.9 CS and SAR PDU formats: (a) AAL3/4; (b) AAL 5.

sequence number (*SN*) is used to detect missing segments. In practice, a device like a network server may have a number of frames – and hence CS-PDUs – being received concurrently from multiple sources. Hence for the receiving SAR protocol to relate each incoming segment to the correct PDU, the source SAR protocol adds the same *message identifier* (*MID*) field to the head of each segment relating to the same CS-PDU. At the tail, the *length indicator* (*LI*) indicates the number of useful octets in the segment since the CS-PDU is not necessarily an integral number of 44 octet segments. Clearly, this field has meaning either in the last segment relating to a multisegment CS-PDU or, if it is the only segment. The *CRC* is used to detect possible transmission errors introduced during the transfer of segments.

AAL 5

Because of the origin of AAL 3/4, the CS-PDU contains a number of fiel the header which were included primarily to support a connection-orie service. The MID field at the head of each SAR-PDU – which is prese enable the destination to relate segments to a specific frame – perfor similar function to the protocol connection identifier – VPI/VCI – prese the header of each ATM cell. As a result, the alternative AAL 5 service was defined. This provides a similar service to AAL 3/4 but with a redu number of control fields in both the CS and SAR PDUs. It is known as **simple and efficient adaptation layer** (SEAL). As with AAL 3/4, its opera is best described by considering the format of the PDUs associated with b the CS and SAR protocols. These are shown in Figure 10.9(b).

As we can see, with AAL 5 there is no header associated with the co structed CS-PDU. Also, the pad field in the trailer is longer (0–47 octets) that the length of each CS-PDU can be an integral number of complete octet segments. This has the advantage that the fields at the trailer of CS-PDU are always the last 8 octets of the last segment, which leads to fas processing at the destination. The AAL *user-to-user identifier* (*UU*) enables two correspondent user layers to relate the AAL SDU to a particular SAP. T use of the *CPI* field has yet to be defined. It is included to support futu functions and currently to make the tail an even number of octets. The *leng* field indicates the number of octets in the user data field and is an integ value in the range 0 to 65 535. The *CRC* field detects the presence of a transmission errors in the reassembled CS-PDU. If errors are detected, th the user layer above is informed when the SDU contained within the PDU delivered. In this way, it is left to the user layer to decide what action to ta in some instances this will be to discard the SDU – for example, if it conta normal data – while in others, it will be to accept it and take appropri recovery steps – for example, if the SDU contents relate to a video or vo portion of a multimedia document.

As we can see, there is no head or tail associated with the SAR PDU. E comprises the full 48 octets and the SAR protocol is said to be null. The of a MID field at the head of each segment means that the segments rela to the same CS-PDU are identified using the field in the header of the A cell that is used to transport the segment. With AAL 5, the SDU type bi user data cells – also known as **ATM-layer user-to-user (AUU) cells** – is used indicate whether the cell contents form the beginning or continuation of CS-PDU (binary 0) or the end (binary 1). Although this means the operati of the AAL layer is now linked with that of the ATM layer, it is done improve the efficiency of the segmentation process. AAL 5 is also used as t AAL layer in the C-plane for the segmentation and reassembly of messag associated with the signaling protocol. It is then known as the **signaling A** or SAAL.

10.4.2 ATM layer

The ATM layer performs all the functions relating to the routing and multiplexing of cells over VCs, which may be semipermanent or set up on demand. Its main function is to assign a header to the segment streams generated by the AAL relating to a particular call. Similarly, on receipt of cell streams, its function is to remove the header from each cell and pass the cell contents – segments – to the appropriate AAL layer protocol.

To perform these functions, the ATM layer maintains a table that contains a list of VCIs. Normally, in ATM LANs for example, all switching is carried out using VPIs only and the VCI field in the cell header is then used to multiplex/demultiplex the cells relating to specific calls/transactions which are transported over the same path. In the case of semipermanent VCs, the VPIs are downloaded by network management – via the network management protocol stack – and in the case of on-demand VCs, the VPIs are obtained using the appropriate signaling protocol set. In both cases the cell streams relating to the management/signaling protocol messages are transferred over permanently assigned VCs.

In a public network, the ATM layer offers various service classes each of which has been specified to meet a particular type of application requirement. These are:

- **constant bit rate (CBR):** this supports isochronous traffic such as uncompressed speech, audio, and video;
- **variable bit rate/real time (VBR/RT):** this supports variable bit rate traffic with real-time requirements. An example is the transmission in real time of a compressed video. There is a declared mean and peak rate and, if the peak rate is exceeded, cells are tagged at the network interface to indicate they can be dropped if necessary;
- **variable bit rate/non-real time (VBR/NRT):** this supports variable bit rate traffic but with no real-time requirements. An example is the transfer of a file containing compressed video from a server to a client workstation;
- **available bit rate (ABR):** this supports bursty traffic with a known minimum, mean, and peak rate. The network guarantees the mean rate but provides the additional rate according to competing demands on the available transmission capacity. If the network is unable to provide the additional rate, however, it informs the user to reduce the input rate;
- **unspecified bit rate (UBR):** this service does not provide any guarantees nor is the user informed if the network is unable to transfer the submitted information. The service offered, therefore, is similar to that provided by a best-effort network such as a LAN or the Internet.

Associated with each service class is a network quality of service (QoS) guarantee which forms the basis of a contract between the network provider and the network user. Essentially, the provider agrees to meet the bandwidth, delay, and delay variation (jitter) requirements of the cell stream associated with a user application providing the cell stream conforms to an agreed set of parameters. A range of QoS parameters have been defined and, for each contract, a set of parameter values is negotiated which the network provider agrees to meet or exceed. Collectively, the parameters form what is called the **traffic descriptor** of the call and, to meet the requirements of the various types of call, a different set of parameters may be defined for the cell stream flowing in each direction of a connection.

The parameters relating to the user include:

- **peak cell rate (PCR):** the maximum rate the source will enter cells;
- **sustained cell rate (SCR):** the average rate the source will enter cells;
- **minimum cell rate (MCR):** the minimum cell rate that is acceptable by the source;
- **cell delay variation tolerance (CDVT):** the maximum level of variation in intercell times.

The parameters relating to the network include:

- **cell loss ratio (CLR):** the maximum ratio of cells that will not be delivered owing to transmission errors or excessive delays within the network;
- **cell transfer delay (CTD):** the average transfer delay of cells across the network;
- **cell delay variation (CDV):** the average variation in cell transfer delay.

In order to enforce the contract, for each cell entered into the network and subsequently delivered, the network determines whether the parameters for this call/VC conform to the agreed contract. The algorithm used is known as the **generic cell rate algorithm (GCRA)** and the principles on which it is based are shown in Figure 10.10.

As we can see, four examples of cell inter-arrival times are given:

(i) Cell 2 arrives according to the agreed PCR and hence is accepted.

(ii) Cell 2 arrives early but within the agreed CDVT. Hence providing the new SCR is not violated, the cell is accepted. Note, however, that the expected time of arrival of cell 3 is $t_0 + 2T$.

(iii) Cell 2 arrives later than the agreed PCR but, since a source can enter cells at a lower rate than the agreed PCR, it also is accepted. Note that the expected time of arrival of cell 3 is the time cell 2 arrived plus T.

(iv) Cell 2 arrives earlier than the agreed CDVT and hence violates the contract. Normally, the network will either set the cell loss priority (CLP) bit in the cell header or discard the cell depending on the current loading of the network.

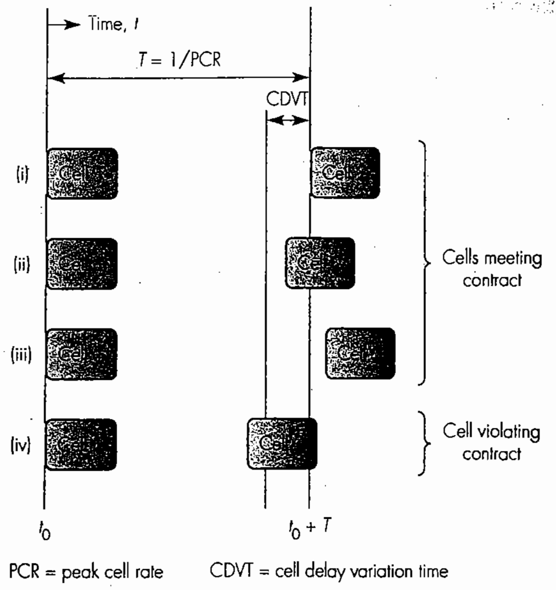
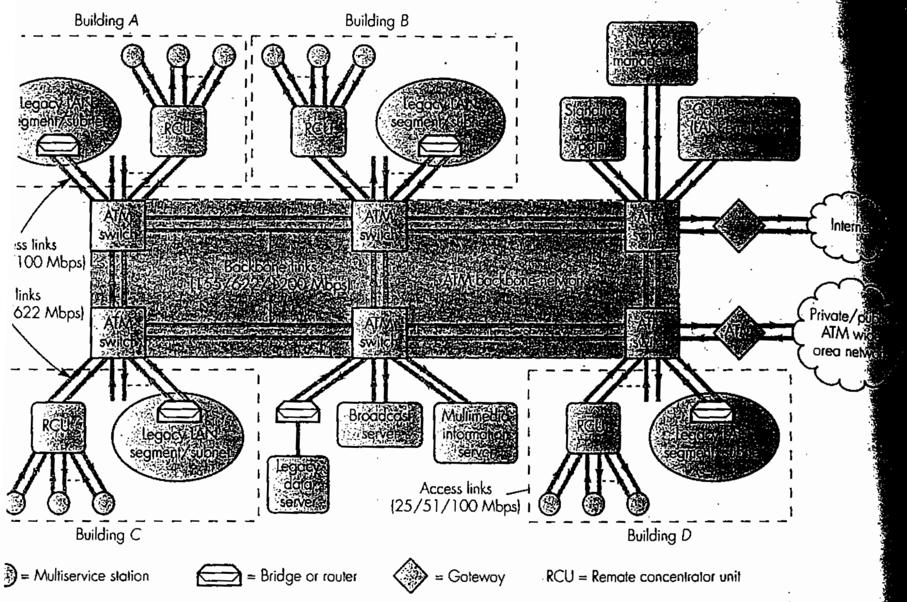


Figure 10.10 Principle of operation of generic cell rate algorithm.

10.5 ATM LANs

ATM LANs have been designed to meet the high bandwidth demands of multimedia applications and hence are an alternative to the LANs based on fast Ethernet which we described in Section 8.7. A typical site LAN is shown in Figure 10.11.

As we explained earlier in Section 10.3, an ATM switch has a defined number of ports and its function is to provide a high bit rate switched communications path between ports. The cost of a switch is a function of the number of ports that it supports. If all the stations are connected directly to the switch then as the deployment of multimedia PCs/workstations increases, switches with a very large number of ports are required. However, not all stations require network services at the same time. To minimize the number of ports, groups of stations – for example, in a building – are connected to the switch through a **remote concentrator unit (RCU)**. There is no switching function in the RCU and its role is simply to multiplex/demultiplex the cell streams from/to those stations that are involved in a network transaction onto/from the link connecting the concentrator to a switch port. In this way, providing the interconnecting link has sufficient capacity to support the



: 10.11 ATM LAN schematic.

anticipated number of concurrent transactions, the number of switch po
required is reduced considerably.

Prior to sending any information relating to a call, a communicatio
path through the network is first established. All the cells relating to the c
are constrained to follow this same path and are delivered in the same ord
as they were submitted. Recall that the conversion of all information i
streams of fixed-sized cells has the advantage that the cells relating to the d
ferent types of call can be switched in a uniform way and independent of t
type of media which they contain. In addition, the use of cells has advanta
in terms of the utilization of transmission bandwidth.

The applications supported by a multimedia PC/workstation inclu
telephony, videophony, conventional data networking, and access to a ran
of related servers. These include, in addition to those present with existi
data networks – electronic mail, printing services, and so on – a broadca
server, a database server supporting multimedia Web pages, and so on. Fo
example, the **broadcast server** – which performs the same functions as a
MCU – enables a user at a multimedia workstation to set up on demand
videoconferencing session between three or more similar workstations b
sending the integrated media streams output from all the workstations
involved directly to the server in real time. The server relays the appropriate
media streams to the other workstations as the conference proceeds.

Similarly, the server holding multimedia Web pages enables a user at a workstation to access a particular page and then interactively browse through it.

In practice, ATM LANs are being introduced in an incremental way and currently a majority of LAN installations are still of the Ethernet type. Within the context of ATM LANs, these are often referred to as **legacy LANs**. The major bottleneck associated with such networks is access to servers, since these require significant bandwidth to support multiple concurrent transactions. Hence in addition to providing direct access to the newer multimedia servers, the backbone switches also provide connections both to the existing data-only LANs in each building – for example through bridges – and to the servers associated with them. Typically, these are all interconnected by a set of point-to-point virtual connections (VCs). In the context of high-speed LANs, the interconnected set of backbone switches can be viewed as providing the same function as a high-speed backbone subnetwork.

All communications across the ATM network are carried out over previously established VCs. These can be set up either on demand by a user or semipermanently by network management. With **on-demand connections**, the user device, before sending any information cells, sends a request for a **switched virtual connection (SVC)** to be set up between itself and the required destination. This can be done either in a distributed way involving all the switches in the connection or using a central control unit known as the **signaling control point (SCP)**. This is responsible for the overall management of both the transmission bandwidth and the setting up and clearing of switched connections through the network. On receipt of the request – known as a **signaling message** – the SCP first determines the availability of both the required destination and the transmission bandwidth appropriate to the call across the network. All the end-user stations are connected to the SCP – normally a powerful workstation – by a separate VC and, assuming the required destination and network resources are available, the SCP sets up routing information in the switch network to link the two user stations (VCs) involved in the call. It then informs the originator of the call that it can commence sending information cells. All the signaling messages associated with the setting up and clearing of calls/connections are transferred across the network to and from the SCP in the form of cells over a separate set of VCs that are permanently set up for this function. The latter are known as **signaling virtual channel connections (SVCCs)**.

To access networked servers such as electronic mail and multimedia information servers, because the ATM network is connection-oriented, before any frames can be transferred, a VC must be in place between each workstation and the set of servers. In practice, the number of servers may be large and this requirement can be met by setting up **permanent VCs (PVCs)** between all user stations and servers and a central data forwarding point known as the **connectionless server (CLS)** or, because it provides a similar service to that provided by a legacy (broadcast) LAN, the **LAN emulation server (LES)**. We shall discuss the operation of this and the SCP in more detail in the next section.

The permanent virtual connections associated with the various services are all set up under the overall control of the network management station. A reserved VC is permanently in place between the **network management station** and the control processor in each RCU and backbone ATM switch. Also betwee all multimedia stations and servers and their network point of attachment. These are used by the network manager to set up the VCs associated with the va ious services. The network manager uses these connections to download routin information for entry into the routing tables held by these devices.

As with existing data-only networks, users of multimedia stations, as wel as needing to communicate with other users at the same site, also want t communicate with users connected to an ATM LAN at a different site. Henc an ATM LAN has a gateway to the Internet and also to newer private/publi wide area ATM networks. To meet these needs, new generations of privat networks based on the ATM are being introduced. Also, a number of telecom providers are introducing a new generation of public network based on the same technology. As we shall see in Sections 10.6 and 10.7, these consist o ATM MANs which, in turn, are interconnected together to form an ATM wide area network.

mple 10.1

A segment of an ATM network is shown in Figure 10.12(a). The numbers alongside each RCU/switch are the port identifiers. Assume that semipermanent VCs are to be set up by network management between stations A, *B*, *C* and *D*, firstly, to the SCP for on-demand calls – for both signaling and information transfer – and secondly, to the CLS/LES for the cell streams relating to connectionless calls. Also assume that a separate VC is required to connect the server to the CLS/LES. Derive typical routing table entries for RCU 1 and RCU 2 and SW 1 and SW 2 to provide these connections assuming VP-only switching is used within the network switches and, within the RCUs, the VPI/VCI field is used to identify specific calls/stations.

Answer

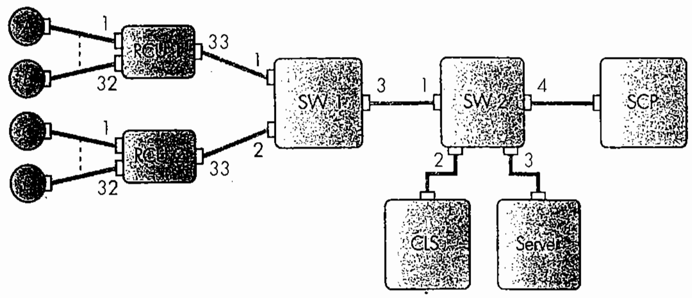
A suitable set of routing table entries is given in Figure 10.12(b). Note the following points when interpreting the entries:

- The SCP, CLS, and server are all connected to their switches by separate transmission lines.
- For on-demand calls, two separate VCs are shown, one between each workstation and the SCP for the signaling messages associated with a call – the signaling channel (SC) – and the other for the cell streams associated with the call – the call channel (CC). Since the latter are to be semipermanent, they are shown set up between each workstation and SW 2. Alternatively they could be set up on demand by the SCP between each pair of workstations involved in a call.
- For connectionless traffic, a separate VC is required between each workstation and the CLS and also between the CLS and the server.
- The SCP, CLS, and server all use the combined VPI/VCI in the cell header to identify the cells relating to specific calls/server transactions.

10.1 Continued

- On the station side of each RCU, the VCI field identifies the port number – and hence station – within each virtual path. Also, in this example, only three virtual paths are required per RCU rather than per station. This allows the approach to be scaled to large installations.
- Within the network, all switching is carried out using VPIs only.
- To set up an on-demand call, the SCP creates entries in the routing table of SW 2 to link the port/VCI of the calling party to that of the called party.
- For connectionless traffic, when relaying the cell streams received from each station to the server, the CLS assigns a new VPI/VCI. Also, in order to relay the cell streams in the reverse direction, it maintains a table that maps the incoming VPI/VCI from the stations to those used for communicating with the server.
- When responding to a request, the server uses the same VPI/VCI values for the cells making up the response as were used in the request.

(a)



(b)

RCU 1:

	In			Out			In			Out		
	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI
SC	1	0	1	33	1	1	33	1	1	1	0	1
CC	1	0	2	33	2	1	33	2	1	1	0	2
CLS	1	0	3	33	3	1	33	3	1	1	0	3
SC	32	0	1	33	1	32	33	1	32	32	0	1
CC	32	0	2	33	2	32	33	2	32	32	0	2
CLS	32	0	3	33	3	32	33	3	32	32	0	3

Figure 10.12 ATM LAN routing example: (a) network segment; (b) example routing table entries.

t.)

RCU 2:	In			Out			In			Out		
	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI
SC	1	0	1	33	1	1	33	1	1	1	0	1
CC	1	0	2	33	2	1	33	2	1	1	0	2
CLS	1	0	3	33	3	1	33	3	1	1	0	3
				⋮						⋮		
SC	32	0	1	33	1	32	33	1	32	32	0	1
CC	32	0	2	33	2	32	33	2	32	32	0	2
CLS	32	0	3	33	3	32	33	3	32	32	0	3

SW 1:	In			Out			In			Out		
	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI
SC	1	1	X	3	1	X	3	1	X	1	1	X
CC	1	2	X	3	2	X	3	2	X	1	2	X
CLS	1	3	X	3	3	X	3	3	X	1	3	X
SC	2	1	X	3	4	X	3	1	X	2	4	X
CC	2	2	X	3	5	X	3	2	X	2	5	X
CLS	2	3	X	3	6	X	3	3	X	2	6	X

X = 1–

SW 2:	In			Out			In			Out		
	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI	Port	VPI	VCI
SC	1	1	X	4	1	X	4	1	X	1	1	X
CC	1	2	X	1	Y	X	1	Y	X	1	2	X
CLS	1	3	X	2	3	X	2	3	X	1	3	X
SC	1	4	X	4	4	X	4	4	X	1	4	X
CC	1	5	X	1	Y	X	1	Y	X	1	5	X
CLS	1	6	X	2	6	X	2	6	X	1	6	X
CLS'	2	10	Z	3	10	Z	3	10	Z	2	10	Z

X = 1
Y = 2
Z = 1

CLS:	In		Out		In		Out	
	VPI	VCI	VPI	VCI	VPI	VCI	VPI	VCI
	3	X	10	Y	10	Y	3	X
	6	X	10	Y	10	Y	6	X

X = 1–32
Y = 2/5

P: Calls in progress

Signaling channel		Call channels						Call type details
		In			Out			
VPI	VCI	Port	VPI	VCI	Port	VPI	VCI	
1	X	1	2	X	1	Y	X	–
4	X	1	5	X	1	Y	X	–

SC = signaling channel
CLS = workstation/CLS channel
CC = call channel
CLS' = CLS/server channel

re 10.12 Continued

10.5.1 Call processing

As we saw in Figure 10.11 and its associated text, there are two types of traffic associated with an ATM LAN: that relating to the message flows exchanged between the distributed community of ATM-enabled stations and their associated servers, and that relating to the message flows exchanged between the bridges/routers that form the interface with the various legacy LAN segments/subnets and their associated servers. Also, in the case of multimedia stations, there are two types of call: one connection-oriented and the other connectionless. The first relates to network services such as telephony and videophony, while the second relates to more conventional data services similar to those used with legacy LANs. We shall consider the three types separately.

Connection-oriented calls

Services such as telephony and videophony require a separate VC to be set up between any pair of workstations for the duration of the call. The standards relating to this type of service are based on those used to set up calls in an ISDN since, in principle, the operation of setting up and clearing calls is the same as that used to perform the same function within an ISDN. All the (signaling) messages relating to the setting up and clearing of calls are carried over a separate channel – called the **signaling virtual channel connection (SVCC)** – which is independent of that used for carrying the message flows associated with the call. When the network is first configured or a new outlet added, a semipermanent VC is set up by network management between each network outlet and the central signaling control point. This is analogous to installing a physical wire connection between each telephone outlet and a local (telephone) PBX.

When an SVCC is set up by network management, an entry is also made in the routing table of the SCP. This consists of the ATM network address of the workstation together with the allocated routing address; that is, the VPI/VCI addresses used in the headers of cells that are received/output by the SCP from/to this workstation. The network address of each station is analogous to that of a telephone outlet and hence is a hierarchical address that uniquely identifies the station in the context of the total network. These have a standard format and, in ATM LANs, are 20-byte addresses.

To initiate the setting up of a call, the user of the station follows a dialog which involves specifying the address of the required recipient of the call and also the call type – telephony, videophone, and so on. This, in turn, results in the exchange of signaling messages using the signaling protocol set in the station and a similar protocol set in the SCP computer. The protocol architecture to support this was shown earlier in Figure 10.6 and the signaling protocol set is defined in **Recommendation Q.2931**. This is based on the ISDN signaling protocol set to ascertain whether the user is prepared to receive a call. If the response is positive, this is communicated back to the SCP which proceeds to set up a (duplex) VC across the network by adding entries into each switch routing table that links the port/VPI/VCI of the calling station with that of the called station. The SCP then informs the cal ing station that a VC has been set up and the call commences. In a simila way, either user can invoke the closing down of the connection at any time b initiating the sending of appropriate disconnection messages over the sep rate signaling channel.

To set up a call involving multiple stations – for example, for a teleconfe encing or videoconferencing session – all the stations must be full interconnected. For small numbers of stations this can be done by the SC directly using a similar procedure to that used for a two-party call. Since i some instances the number of stations involved may be large, an alternativ approach is to route all the information flows relating to such calls via a cen tral routing point called the broadcast server.

Using this approach, to set up a conferencing session, the initiator of th session communicates with the SCP as before – using the signaling channel but this time indicates a conferencing call – and its type – and the addresse of the stations involved. The SCP, in turn, communicates with the user o each of these stations – using their signaling channels and associated proto col set – to ascertain their availability and willingness to participate in conferencing session. On receipt of their responses, the SCP initiates the se ting up of a switched VC between each station that returns a positive response and the broadcast server. It then informs, firstly, the broadcast server of the type of call and the set of switched VCs involved, and secondly, the users o the stations that the session can commence. The session then proceeds wit the broadcast server relaying the information flows received from each sta tion out to all the other stations. These play out the received informatio under the control of the user of the station, for example, by using a separate window for each member of a videoconferencing session.

Connectionless calls

As we mentioned earlier, ATM-enabled stations are often introduced in a incremental way as existing frame-based LAN segments are replaced by cel based (ATM) segments. A major issue when considering connectionles working is interworking, not only between ATM (cell-based) stations, but als between such stations and stations that are connected to legacy-LAN segment

As we explained in Chapter 8, most large legacy LANs consists of number of LAN segments of the same type interconnected by bridges Alternatively, if the segments are of different types, as we saw in Chapter 9 they are interconnected by subnet routers. Recall that when using bridges a routing is carried out at the MAC sublayer. When using routers, howeve routing is carried out at the IP/IPX layer. Both the MAC and IP layers pro vide a best-effort connectionless service for the transfer of preconfigure frames or datagrams, respectively. To support interworking between a station with a cell interface and a station connected to a legacy LAN, two alternatives must be supported: one for use with a bridged LAN and the other for use with a router-based LAN. With the first, the interface between the ATM LAN

and the legacy LAN is a bridge and with the second the interface is a subnet router. Two different protocol architectures are used, each of which provides seamless interworking in the appropriate environment. We shall consider each separately.

LAN emulation

The method used when bridges form the interface to the legacy LAN has been developed by a group of companies that are all involved in the manufacture of LAN networking equipment. This group is called the **ATM Forum**. The various networking components used with this method are shown in Figure 10.13(a).

The aim of the various networking components is to emulate the broadcast mode of operation of a legacy LAN over a connection-oriented ATM LAN. The term used to describe this method is known as **LAN emulation (LE)**. The three components used are an **LE configuration server (LECS)**, an **LE server (LES)**, and a **broadcast and unknown-address server (BUS)**. Although each is shown as a separate entity in Figure 10.13, for a small LAN, they may all be implemented in a single computer. In this case, the message/frames relating to each component are identified by the type of virtual channel connection (VCC) on which they arrive.

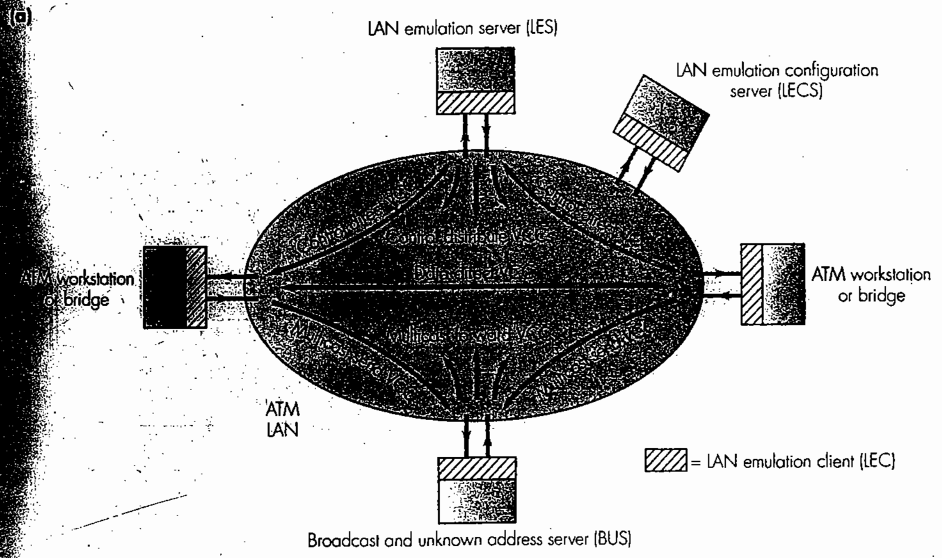
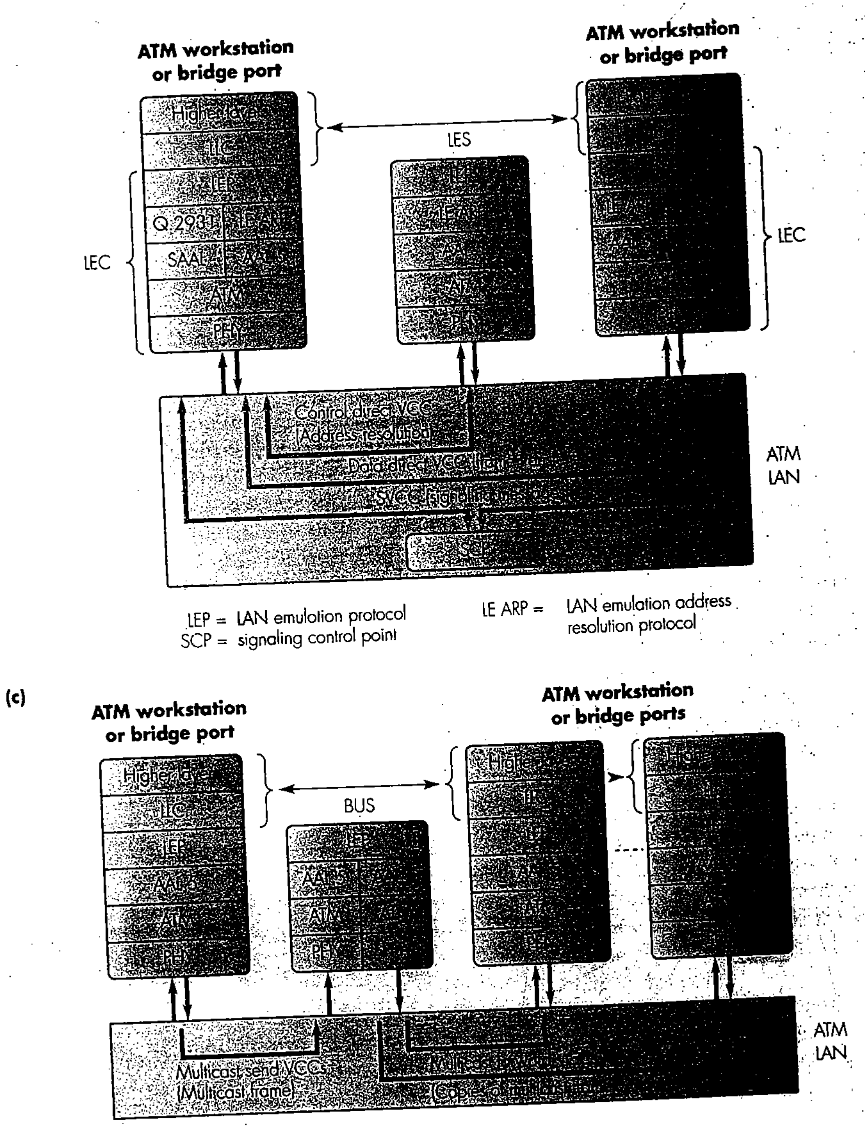

(a)
LAN emulation server (LES)
LAN emulation configuration server (LECS)
Control distribute VCC
Data direct VCC
Multicast forward VCC
ATM workstation or bridge
ATM workstation or bridge
ATM LAN
▨ = LAN emulation client (LEC)
Broadcast and unknown address server (BUS)

Figure 10.13 LAN emulation: (a) terminology and networking components; (b) unicast protocol architecture; (c) multicast protocol architecture.


ATM workstation or bridge port
ATM workstation or bridge port
LES
Higher layer
LLC
LEP
Q.2931
LE ARP
SAAL
AAL5
ATM
PHY
LEC
LEC
Control direct VCC
(Address resolution)
Data direct VCC
SVCC signalling
SCP
ATM LAN
LEP = LAN emulation protocol
SCP = signaling control point
LE ARP = LAN emulation address resolution protocol
(c)
ATM workstation or bridge port
ATM workstation or bridge ports
BUS
Multicast send VCC
(Multicast frame)
Multicast forward VCC
(Copies of frame)
ATM LAN

gure 10.13 Continued

In a large installation there may be multiple emulated LANs each of which has its own LES and BUS. The LECS is used by all stations that are connected to the total ATM LAN to determine the ATM addresses of the LES and BUS. There is a reserved VCC to connect all stations to the LECS for this purpose. The LES provides an address resolution service to convert from 48-bit MAC addresses into 20-byte ATM addresses. The BUS is responsible, firstly, for supporting multicasting/broadcasting, and secondly, for relaying frames to stations whose MAC address is unknown by the LES.

All stations connected to the ATM LAN – ATM-enabled stations, bridges, and so on – have an **LE client** (**LEC**) subsystem associated with them. This comprises hardware and software that provides a similar service to the MAC chipset (and associated software) in a station that is connected to a legacy LAN. When a station is powered up, it goes through an initialization procedure at the end of which it has initialized certain operational parameters and its own set of addresses – as described below, a station has more than one address. As we can see in Figure 10.13(a), a separate pair of VCCs is used to connect each LEC to the LECS, the LES, and the BUS. Once initialized, the LEC first obtains the pair of VCCs of the LES and BUS from the LECS and then proceeds to register itself with both the LES and BUS. At the end of the initialization phase, both the LES and BUS have the set of addresses of all active stations that are connected to the ATM LAN. In the case of a bridge, the LEC can either register the MAC addresses of all stations that are connected to its legacy LAN port or, more usually, just the address of the port through which it is connected to the ATM LAN. In the first case the bridge operates in the **nonproxy mode** and in the second, the **proxy mode**.

We can see in Figure 10.13(b) that the protocol stack within each LEC contains a **LAN emulation protocol** (**LEP**) layer – immediately below the LLC sublayer – which communicates with a similar layer in the LES. In order to make the underlying network transparent to the LLC sublayer, the service provided by the LEP layer is a connectionless service similar to that offered by the MAC sublayer. The two user service primitives are LE_UNITDATA.request and LE_UNITDATA.indication.

To emulate a broadcast LAN, both primitives have a source and destination MAC address as parameters. This means that each workstation and bridge connected to the ATM LAN, in addition to a 20-byte ATM address, has a 48-bit MAC address associated with it. It also has a 2-byte **LEC identifier** (**LECID**) which is used to identify uniquely the ATM workstation or bridge port among those currently attached to the LES. We shall look at the function of the LECID later.

The VCC that (logically) connects an LEC to the LES is called the **control direct VCC**. On receipt of an LE service request primitive, the LEP in the source LEC reads the source and destination MAC addresses and passes these to the **LE address resolution protocol** (**LE ARP**). The latter forms an address resolution request message – containing the two MAC addresses and the ATM address of the LEC – and sends this to the LE ARP in the LES over the control direct VCC. Assuming the LE ARP in the LES knows the ATM addres of the destination LEC, it returns this in a reply message to the requesting LE ARP – the latter initiates the setting up of a **data direct VCC** between itsel and the LE ARP in the destination LEC. This is done using either the SC and the signaling protocol set described earlier for connection-oriented cal or, with some ATM switch architectures, directly by the two LE ARPs involved The connection is cleared by the LE ARP when no further frames ar received for the same destination LEC within a defined timeout interval.

If the LE ARP in the LES does not have the ATM address of the destin tion LEC – for example, if the destination MAC address relates to a station connected to the legacy LAN port of a bridge – the LES sends a copy of the LE ARP request message to all registered LECs using the set of **control di tribute** VCCs. The LE ARP which has knowledge of the destination MAC address then replies with the corresponding ATM address – or its own ATM address if it is a bridge – using the control direct VCC. Once the source LE ARP has obtained the ATM address of the destination, it sets up a data direc VCC as before.

The flow of data frames, each of which has the source and destination MAC addresses at its head, can then start. Since bridges are involved these can be either Ethernet, 802.3 or 802.5 frames with the frame type defined in the header of the frame. No FCS field is required since, in an ATM LAN, this is performed by the AAL 5 sublayer. Because of the use of the inter mediate address resolution phase, the LE connectionless service is said to be provided *indirectly*.

The foregoing relates to unicasting, that is, there is a single destination station for each submitted MAC frame. To implement multicasting, a copy of the created MAC frame must be forwarded to all stations that belong to the same multicast group. Clearly, using the method just described, a multiplicity of switched VCs would be required between each member of the group and all other members. To avoid this requirement, the BUS is used and the assoc ated protocol architecture is as shown in Figure 10.13(c).

An additional pair of VCCs is used between the LEC in each station and the LEC in the BUS. In the station-to-BUS direction, the VCC is known as the **multicast send VCC** and in the reverse direction, the **multicast forward VCC**. On receipt of a service primitive with a multicast destination address, the LE creates a frame as before but with its own 2-byte LECID at the head. It sends this directly to the LEP in the BUS over the multicast send VCC. On receip of the frame, the LEP broadcasts a copy of the frame to all stations using the set of multicast forward VCCs. The LEP in each station first determines from the LECID at the head of the frame whether it originated the frame. If it did then the LEP simply discards the frame. This is known as **echo suppression.** I the LECID does not match that of the station, the receiving LEP determines from the multicast address at the head of the frame whether this station is part of the multicast group. If it is, the frame is passed up to the LLC layer, if not, the frame is discarded.

The unknown address service associated with the BUS enables an LEP to send a limited number of frames to their intended destination during the period a data direct VCC is being set up. The procedure followed is the same as that for a multicast frame and, since multiple copies of each frame are sent – each over a separate multicast forward VCC – a limit is set on the number of such frames the LEP can send. Any further frames received must then be retained – cached – until the data direct VCC is in place.

Classical IP over ATM

The method used when subnet routers form the interface to the legacy LAN has been developed by the Internet Engineering Task Force (IETF). The basic method is known as **classical IP over ATM (IPOA)**.

Recall from Chapter 9 that, using the IP, before a station/host can exchange datagrams with another station, it must first know the IP/MAC address-pair of its local subnet router. Also, the local subnet router must know the address-pair of all the stations that are connected to the LAN segments to which it is attached. These are acquired using the ARP protocol and exploiting the broadcast nature of legacy LANs. Once these have been acquired, all datagrams/packets are exchanged between two stations via the router and it is this that relays the packets to their intended destination using the MAC address corresponding to the IP destination address in the datagram header.

To emulate the same operation with a non-broadcast ATM LAN, a permanent VCC is set up between all stations and router ports that are connected to the ATM LAN and a central node known as the connectionless server (CLS). The CLS provides a dual function of address resolution and a datagram/packet relaying service. Since with this method all packets are relayed directly by the CLS, the connectionless service is said to be provided *directly*. The protocol architecture used with a CLS is shown in Figure 10.14.

The ARP in each station/router port first registers its own address-pair – IP and ATM addresses – with the ARP in the CLP using the corresponding permanent VCC. In a similar way, the ARP in the CLS can acquire the address-pair of a station or router port that is connected to it by a VCC. In this way, the ARP in the CLS builds up a routing table containing the address-pair of all stations and router ports that are connected to it. Whenever a station IP has a datagram to send, it simply sends the datagram to the IP layer in the CLS over the VCC that connects the station to the CLS. On receipt of the datagram, the IP in the CLS first reads the destination IP address from the head of the datagram and uses this to obtain the corresponding ATM address from its routing table. The IP then forwards the datagram/packet to the destination over the corresponding VCC.

Multicasting with a CLS is carried out in a similar way to that described with an LES except with a CLS a separate broadcast server is not required. Instead, on receipt of a datagram with a multicast address, the IP layer in the

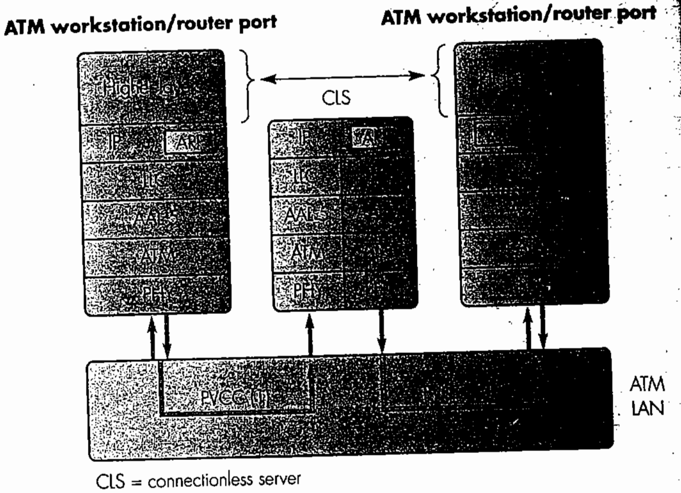

ATM workstation/router port
ATM workstation/router port
CLS
Higher layers
IP
ARP
LLC
AAL5
ATM
PHY
IP
LLC
AAL5
ATM
PHY
PVCC(1)
ATM LAN
CLS = connectionless server

Figure 10.14 Protocol architecture to support classical IP over an ATM LAN.

CLS simply sends a copy of the datagram to all other stations using their co
responding VCC. Each recipient IP then determines from the multica
address at the head of the datagram whether it is a member of the multica
group. If it is, the datagram is passed to the higher protocol layers, if not, th
datagram is discarded.

When a CLS is being used, there are two alternative relaying modes. Wi
the first, the complete datagram is reassembled by the AAL before it
processed by the IP layer. Clearly, this introduces delays and requires sign
cant buffer storage for its implementation. In order to reduce the
overheads, we may use a second mode of working known as **streaming**
pipelining. In practice, the destination address required for routing is car
in the first ATM segment/cell and hence in this mode, the first (beginnin
message) segment is passed directly to the IP layer for processing as soon a
is received by the AAL protocol. The new VC is then determined and this a
the remaining segments making up the datagram are relayed directly witho
reassembly. The AAL with this method is AAL 3/4 since the MID field in t
header of each cell is required for identification purposes during th
reassembly process.

10.6 ATM MANs

Metropolitan area networks (MANs) have been installed by a number of network providers to meet the demand from customers for a switched high-speed LAN interconnection facility. Prior to the introduction of MANs, the only way of interconnecting a set of LANs that belong to different enterprises was by means of high bit rate leased lines. The disadvantage of this approach is the large number of leased lines that are required and, for each additional site, a new set of lines must be put in place. Initially, the LANs were all distributed around a large town or city but later the service was extended to cover multiple towns and cities.

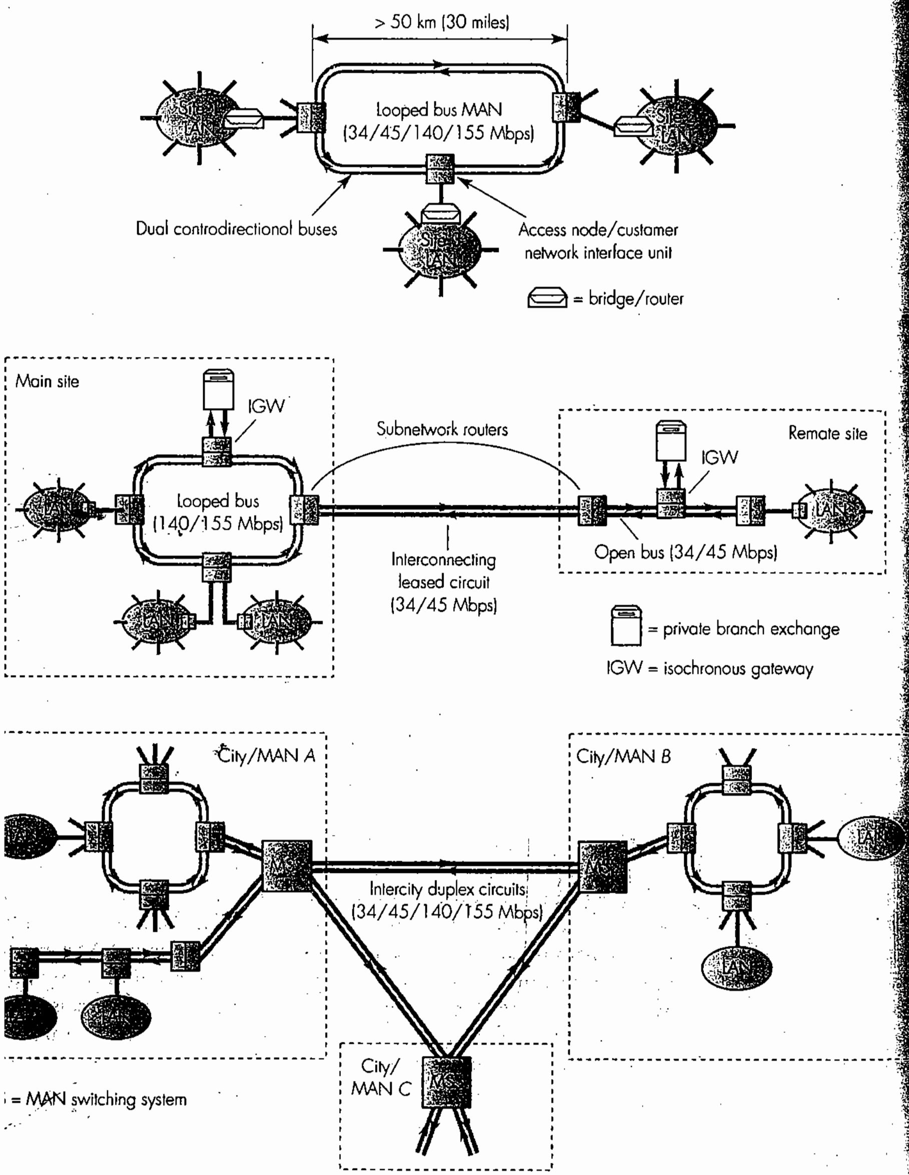
All MANs operate using the asynchronous transfer mode and hence one approach to implementing a MAN is to use a set of interconnected ATM switches similar to the ATM backbone network we showed earlier in Figure 10.11. An alternative (and often less costly) solution is to use a high bit rate shared transmission medium. This is the approach taken in **distributed queue dual bus** (**DQDB**) MANs and, since this has been adopted by a number of providers, we shall limit our discussion of MANs to DQDB.

DQDB is an international standard that is defined in **IEEE 802.6**. The standard relates to a single subnetwork in the context of a larger network; typically, the larger network comprises a number of interconnected DQDB subnetworks. Each subnetwork is made up of **dual contradirectional buses** – that is, two unidirectional buses running in opposite directions – to which a distributed community of **access nodes** – also known as **customer network interface units** – are attached. The buses may be in an **open bus topology** or a **looped bus topology**, in which case the two ends are separate but physically colocated. As we shall see later, this can be used to provide better fault tolerance. Three example applications are shown in Figure 10.15: part (a) shows a single subnetwork MAN; part (b) a typical split-site private network; and part (c) a large wide area public network comprising multiple interconnected MANs.

As part (a) shows, a single DQDB MAN subnetwork can cover an area in excess of 30 miles (50 km). In this example, the access nodes may be located at the telephony switching offices in that area. The use of 34/45/140/155 Mbps circuits provides **seamless interconnection** for many LANs distributed around the city; that is, two users at different sites communicating with one another will be unaware of the fact that an intermediate network is involved.

Part (b) illustrates the use of two DQDB subnetworks in a private network application. In this instance, all the networking equipment located at both sites are privately owned and run and the high bit rate circuit interconnecting both sites leased from a public carrier. One site is assumed to have a looped bus topology and the other an open bus. This example also shows how a DQDB network can be used for the interconnection of two private telephone exchanges (PBXs) which requires a constant bit rate channel.

Part (c) illustrates how several multisubnetwork MANs can be interconnected together to form a larger network that spans several cities. As we can


> 50 km (30 miles)
Looped bus MAN
(34/45/140/155 Mbps)
Dual contradirectional buses
Access node/customer
network interface unit
= bridge/router
Site LAN
Main site
IGW
Subnetwork routers
Remote site
Looped bus
(140/155 Mbps)
Interconnecting
leased circuit
(34/45 Mbps)
Open bus (34/45 Mbps)
LAN
= private branch exchange
IGW = isochronous gateway
City/MAN A
City/MAN B
City/
MAN C
MS
Intercity duplex circuits
(34/45/140/155 Mbps)
= MAN switching system

e 10.15 DQDB/MAN network architectures: (a) single-site MAN; (b) dual-site private ork; (c) wide area multiple MAN network.

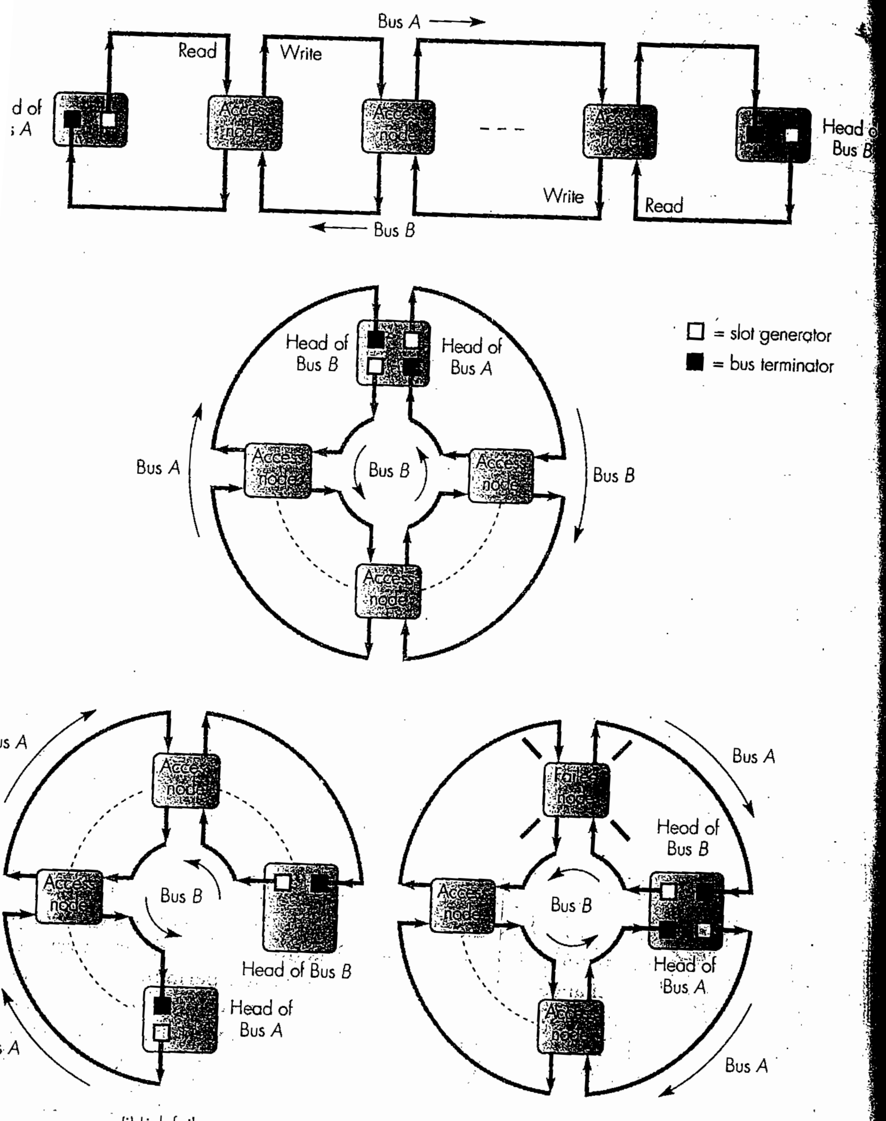
see, in this case each DQDB subnetwork in a city is connected to a **MAN switching system** (MSS). The MSS, on receipt of each LAN frame, either relays it directly to the required subnetwork – if it is connected to it – or it forwards the frame to the required MSS over the appropriate intercity duplex circuit. Such a network provides a **switched multimegabit data service** (**SMDS**) similar in principle to that provided by a frame relay network.

Normally, a LAN is connected to its nearest access node by either a remote bridge or a router. All frames are transported across each DQDB subnetwork in the form of fixed-sized units known as segments. At the interface to each subnetwork, the frames are first fragmented into segments and then reassembled back into their original form at the destination. Because the format of LAN frames differs from one type of LAN to another, prior to the transfer of a frame, a standard header and trailer are added to it. With public networks, the header contains two new source and destination addresses which are the networkwide addresses of the corresponding customer access/interface units. As we shall see in Section 10.6.7, these are hierarchical addresses which identify the MSS, the subnetwork, and the customer access unit. All segments are transferred over each subnetwork using both buses which, because these pass data in opposite directions, ensures that a copy of each segment transmitted is received by all nodes on the subnetwork. Between subnetworks, routing is carried out on reassembled frames using the networkwide destination address at the head of each frame.

10.6.1 Subnetwork architectures

A schematic diagram of an open dual-bus subnetwork is shown in Figure 10.16(a). At the head of each bus is a **slot generator** which generates a contiguous stream of **slots** each capable of transferring a standard 53-octet cell. Access nodes are attached to both buses through read and write connections, with the read operation performed ahead of the write operation. The physical layer in each access node reads the contents of each slot without modifying its contents, and only if new data is to be sent by an access node does it overwrite the existing contents. We can conclude, therefore, that an access node only copies (reads) data from a bus, it does not remove it. Also, it only changes the contents of a slot when it is permitted to do so by the access protocol. Hence failure of the access unit within a node does not have any effect on the contents of the slots that pass by on the two buses providing the failure does not cause the node continuously to write.

A looped bus architecture is shown in Figure 10.16(b) and, as we can see, in this configuration both slot generators are colocated. Alternatively, the same slot generator can be used for both buses. In both cases, the two buses are still independent and are not connected together as in a ring network. Also, in addition to generating slots, the **head of bus** is responsible for passing management information to the access nodes. Typically, management information is generated by a separate network management station and



(i) Link failure (ii) Node failure

.16 DQDB architectures: (a) open bus; (b) looped bus; (c) example reconfigured s networks.

relates to either the allocation of isochronous bandwidth or the operational integrity of the subnetwork.

With a looped bus topology reconfiguration can be supported in the event of a link or a node failure by replicating the head of bus functions so that more than one node can assume the role of head of bus. Two examples of reconfigured loops are shown in Figure 10.16(c); the first illustrates the reconfigured loop after a link failure and the second the reconfigured loop after the failure of an access node. The reconfiguration operation is carried out under the control of a remote network management station.

10.6.2 Protocol architecture

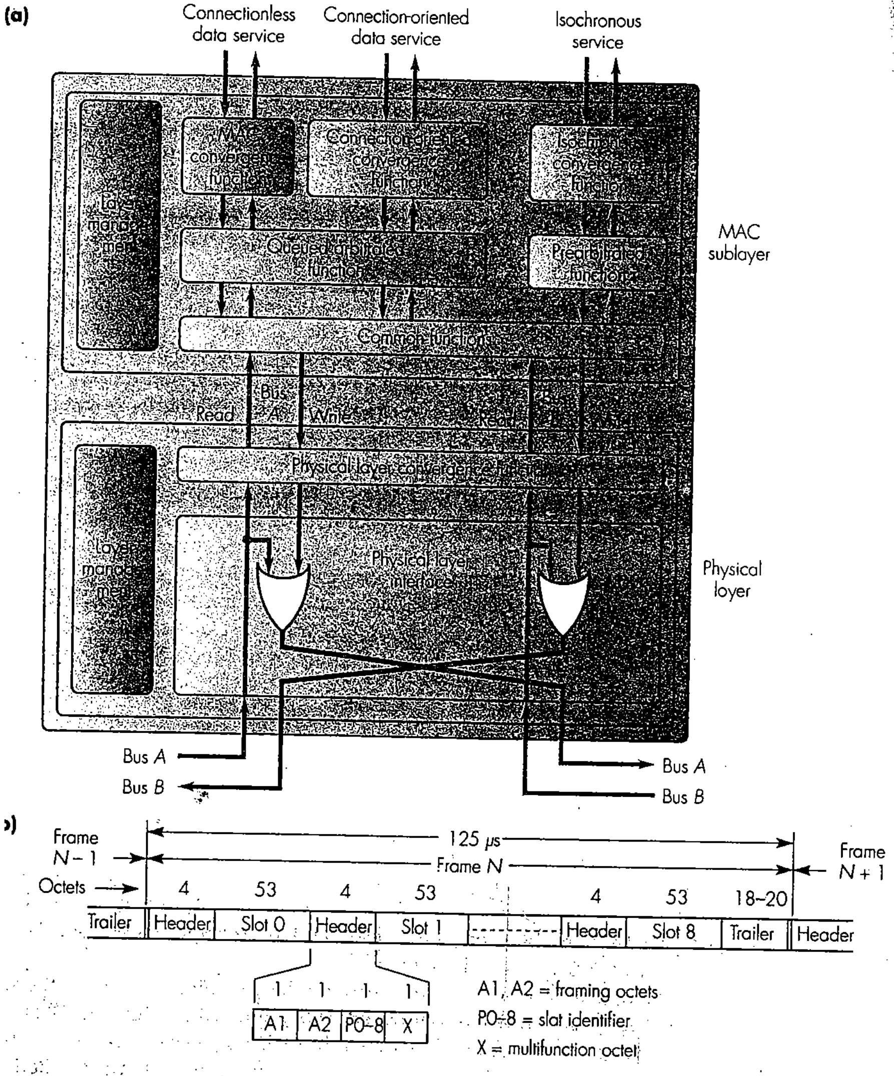
Figure 10.17(a) identifies the components of the protocol architecture defined in IEEE 802.6. As with the other IEEE 802 standards, this defines the operation of the MAC sublayer – known as the DQDB layer in the standard – and the physical layer.

In addition to the normal connectionless (best-effort) data service provided for the interconnection of the frame-based LAN types, the DQDB MAC sublayer provides two additional services: a connection-oriented (reliable) data service and an isochronous service. Typically, the latter is used for the interconnection of two private (telephone) branch exchanges. For such applications, the multiplexed voice samples within the duplex link connecting the exchanges to their access nodes must be transferred across the bus at the same rate as the access link, hence the term "isochronous".

MAC sublayer

As Figure 10.17(a) shows, the MAC sublayer comprises four main functions: convergence, bus arbitration, common, and layer management. As we indicated earlier, all information is transferred across the dual buses in the form of fixed-sized segments. The convergence functions are required to convert all incoming source information into segments before its transfer and back into its original form before its delivery. For example, the MAC convergence function fragments the data frames submitted by a bridge or router into multiple segments ready for their transfer and, on receipt, reassembles them back into frames prior to their delivery.

There are two alternative control modes for gaining access to the slots on the two buses: **queued arbitrated** (QA) and **prearbitrated** (PA). When isochronous services are supported, in order to provide a constant bit rate service, the required number of slots are preallocated and marked by the head-of-bus node. The prearbitrated function in each node providing the service is responsible for identifying the reserved slots relating to it – from an identifier at the head of each slot – and then either initiating the transmission of the isochronous data within these or receiving the data contained within them. All the remaining bus slots are used for the transfer of LAN information frames. Access to these slots is controlled by a distributed queuing algorithm in the queued arbitrated function block.


(a)
Connectionless data service
Connection-oriented data service
Isochronous service
MAC convergence function
Connection-oriented convergence function
Isochronous convergence function
Layer management
Queued arbitrated functions
Prearbitrated functions
MAC sublayer
Common functions
Bus A
Read
Write
Physical layer convergence function
Physical layer interface
Layer management
Physical layer
Bus A
Bus B
Bus A
Bus B
Frame N − 1
125 µs
Frame N
Frame N + 1
Octets
4
53
4
53
4
53
18–20
Trailer
Header
Slot 0
Header
Slot 1
Header
Slot 8
Trailer
Header
1
1
1
1
A1
A2
P0–8
X
A1, A2 = framing octets
P0–8 = slot identifier
X = multifunction octet

e 10.17 DQDB protocol architecture: (a) layer functions; (b) example physical layer rgence function.

We can see from Figure 10.17(a) that the common function block forms the interface between the physical layer and the two bus arbitration function blocks. The physical layer interface takes the form of single octet transfers – read and write – to and from both buses. The main role of the common function block is to relay octets between the physical layer and the appropriate bus arbitration function. This involves detecting the start and end of each slot and examining selected octets within the slot/cell header according to its defined format. We shall consider this in more detail in Section 10.6.6. In addition, since more than one node can act as the head of bus (to enhance reliability) the nodes which can perform this function take part in the reconfiguration operation of the bus/loop when faults are detected. Also, when selected as the head of bus, the node generates contiguous streams of slots. Both these operations form part of the common functions.

Physical layer

The physical layer provides a standard interface to the MAC sublayer. However, as Figure 10.15 showed, the physical layer can be implemented using a range of different transmission media. For example, with public networks this can be 34, 45, 140, or 155 Mbps. As we saw in Sections 7.2.3 and 7.2.4, such circuits utilize different framing formats and so a **physical layer conversion function** – which establishes a contiguous sequence of slots over the selected physical transmission medium – must be provided. An example of how this is achieved is shown in Figure 10.17(b).

The example relates to the use of a 34.368 Mbps E3 digital circuit. As the numbering indicates, this is at the third level in the PHD multiplexing hierarchy and is used to transmit four 8.448 Mbps streams. As we showed in Figure 7.17, a number of bits/octets are used for framing and other purposes and hence a lower bit rate is available for the transmission of DQDB slots. When using such circuits for each bus, the head of bus establishes the frame structure shown in Figure 10.17(b) using the available transmission bandwidth. This has a duration of 125 µs which ensures that the frame can be used to support various constant bit rate voice services as well as data transfer. As we can see, preceding each 53-octet slot is a 4-octet header. The first two octets of the header enable the physical layer convergence function in each node to synchronize to the start of each new frame. The third octet identifies the slots within each frame. The fourth octet is used for a number of functions including the transfer of management information from the head of bus to the two layer management entities in the attached nodes. The physical layer convergence function reads and interprets the octets in the header field but only passes the 53-octet cell in each slot to the MAC sublayer. Because of the fixed 125 µs interval, each frame does not contain an exact multiple of 57 octets. The unused octets in this field arise due to the plesiochronous (nearly synchronous) mode of operation of such circuits as described in Section 7.2.4. With higher bit rate circuits, the 125 µs interval is maintained and hence each frame contains proportionately more (DQDB) slots.

Finally, as with all the protocol layers, the layer management entities a
ciated with the physical layer and MAC sublayer are provided for a rem
management entity to control and monitor the operation of both th
layers. For the MAC sublayer this involves receiving and acting upon infor
tion relating to the slots, for example, the list of identifiers that are carrie
the head of each prearbitrated slot and the message identifiers that are at
head of queued arbitrated slots, the use of which we shall describe in Secti
10.6.6. It also involves the setting of operational parameters such as tim
and the activation of the configuration control procedure. All informati
relating to such operations is carried in the header that precedes each slot

10.6.3 Queued arbitrated access protocol

Access to the slots that are available for the transfer of asynchronous dat
based on a distributed queuing algorithm. This is known as **queued-pac**
distributed-switch (QPSX) and the principles of the method are shown
Figure 10.18.

When interpreting the figure, it should be remembered that for an acc
control unit (ACU) to broadcast a segment/cell to all other ACUs, it m
send a copy of the segment on both buses. Also, each bus operates indep
dently and the QPSX algorithm ensures each ACU gains access to both bu
in a fair way.

As part (a) shows, each slot contains two bits at its head that are used
the distributed queuing algorithm: the **busy** or **B-bit** and the **request** or **R-**
Also, as part(b) shows, access to the slots on each bus is controlled by a se
rate **request counter**. To ensure access to empty slots on each bus is f
requests for slots on one bus are made using the R-bit in slots on the ot
bus. Then, for each counter, whenever a slot passes with the R-bit set, the co
tents of the corresponding counter are incremented by one. Simila
whenever an empty slot is repeated at the interface of the opposite bus, t
counter is decremented by one. At any point in time, therefore, the requ
counter for the corresponding bus contains the number of outstan
requests for slots on that bus from the access nodes that are downstrea
the access node. Note that since there can be a different number of ACU
each side of the access mode, each bus operates independently of the oth

Figure 10.18(c) shows that, to transmit a cell (containing a segmen
data) on bus *A*, the sending access node transfers the current contents o
request counter for bus *A* to a second counter known as the **countdo**
counter. The node then resets the contents of the request counter to z
and sets the R-bit in the first slot received on the opposite bus with the R-b
reset. This has the effect of placing the segment into the distributed que
for bus *A*. The same procedure is followed concurrently with the set of cou
ters associated with bus *B* to place a copy of the segment in the distribut
queue for bus *B*.

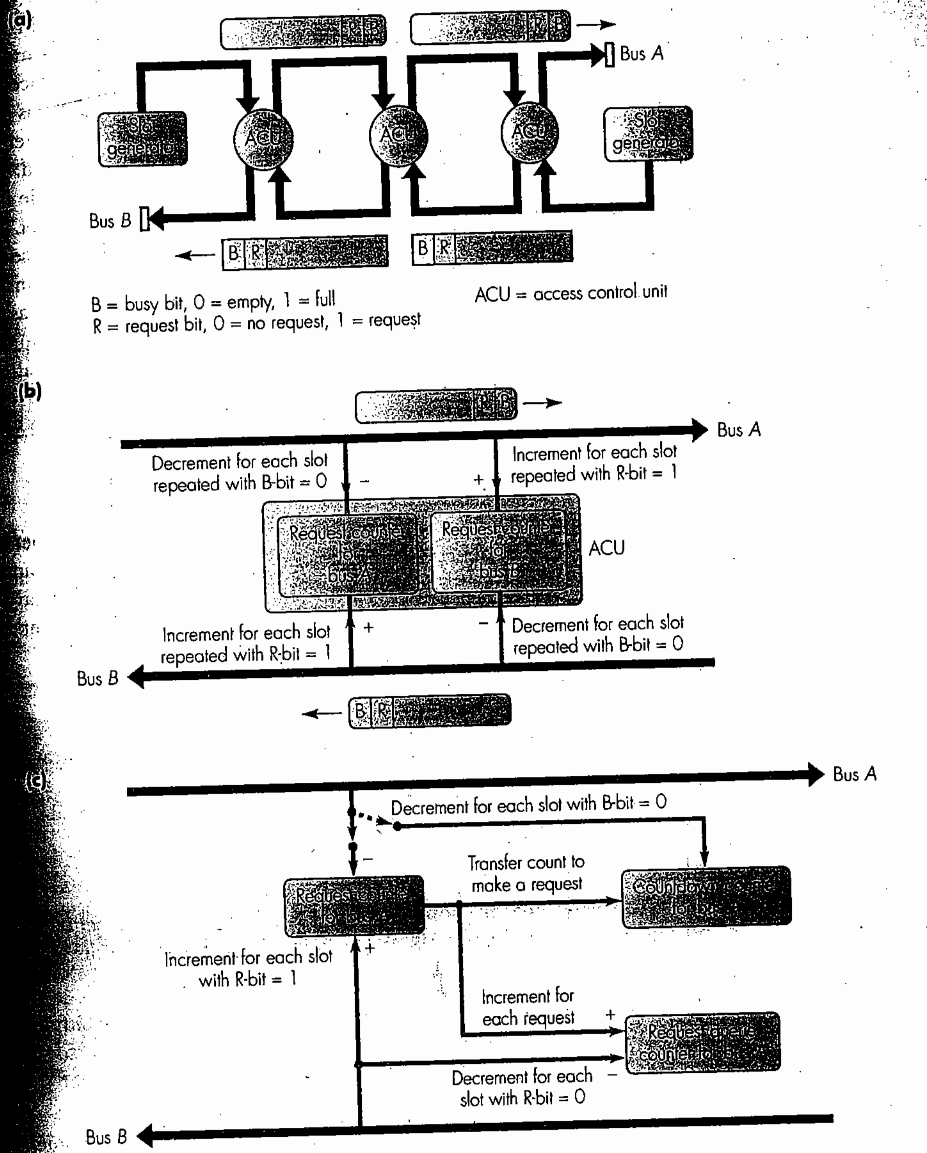

(a)
B R
Bus *A*
Slot generator
ACU
ACU
ACU
Slot generator
Bus *B*
B R
B = busy bit, 0 = empty, 1 = full
R = request bit, 0 = no request, 1 = request
ACU = access control unit
(b)
B R
Bus *A*
Decrement for each slot repeated with B-bit = 0
Increment for each slot repeated with R-bit = 1
Request counter for bus A
Request counter for bus B
ACU
Increment for each slot repeated with R-bit = 1
Decrement for each slot repeated with B-bit = 0
Bus *B*
B R
(c)
Bus *A*
Decrement for each slot with B-bit = 0
Transfer count to make a request
Countdown counter for bus A
Request counter for bus A
Increment for each slot with R-bit = 1
Increment for each request
Request queue counter for bus B
Decrement for each slot with R-bit = 0
Bus *B*

Figure 10.18 DQDB access control principles: (a) request/busy bits; (b) request counters; (c) queuing mechanism.

While a segment is queued for a bus, any slots that are repeated with th R-bit set cause the corresponding request counter to be incremented a before. However, slots which are repeated at the opposite bus interface wi their B-bit reset (zero), cause only the countdown counter for that bus to b decremented. The queued segment is then transmitted on the bus when th corresponding countdown counter is zero and an empty slot is received.

In practice, since both buses operate independently, it is possible fo string of slots to be received on the opposite bus all with their R-bits set. Th means that it may not be possible to set the R-bit in a slot – in response to new request – until more than one segment has been queued and transmi ted on the opposite bus. To allow for this, a third counter known as th (local) **request-queue counter** keeps a record of outstanding request Whenever a new request is made, the counter is incremented and, all th time the counter contents are greater than zero, whenever a slot is receive with its R-bit reset, the R-bit is set and the counter is decremented.

10.6.4 Bandwidth balancing

Following the introduction of the draft standard of DQDB, detailed perfor mance analyses of the queued-arbitrated protocol demonstrated that, unde heavy load conditions, the nodes nearest to the head of each bus start to obtain preferential access to both buses relative to the nodes nearer th centre of the bus. We can best understand the reason for the unfairness by remembering that the node at the head of each bus has first call on the use of the request bit in the slots that pass on one of the buses. Also, although the same node is the last to make requests on the other bus, the related empty slots pass this node first. Under heavy load conditions when the demand for

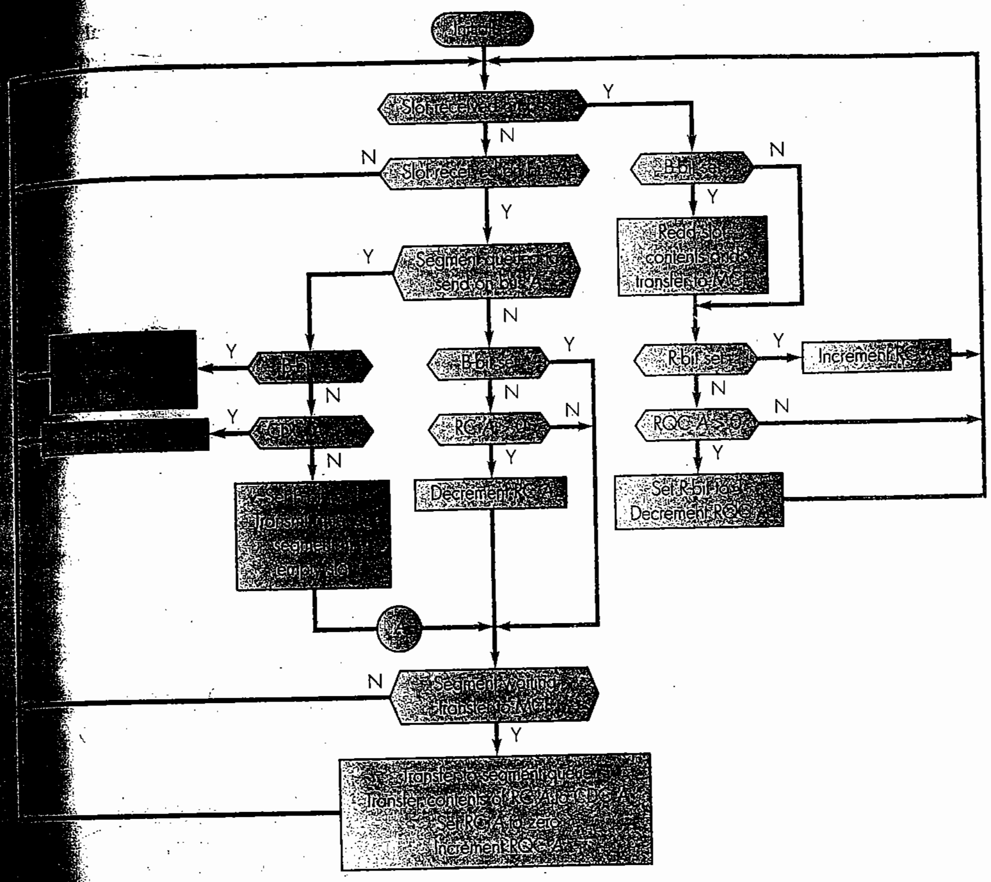
ample 10.2

Derive a flowchart showing the steps taken by the queued-arbitrated function to effect the transmission of a set of queued segments provided by the MAC convergence function on a single bus of a dual-bus DQDB subnetwork.

Answer:

A flowchart showing the steps to control the transmission of segments on bus A is given in Figure 10.19. Consider the following points when interpreting the figure:

- On receipt of a full slot on either bus, the queued-arbitrated function simply passes the contents of the slot payload directly to the MAC convergence function. This function determines whether the segment is intended for this node.
- Only a single segment can be queued for transmission by the queued-arbitrated function at one time. Hence only after this function has transmitted a segment does it return to the output queue of the MAC convergence function to determine whether another segment is awaiting transmission.


Initialize
Slot received on bus B?
Slot received on bus A?
Y
N
B-bit set?
Read slot contents and transfer to MCF
Segment queued to send on bus A?
B-bit set?
CD A
Transmit queued segment in empty slot
RC A > 0?
Decrement RC A
R-bit set?
Increment RC A
RQC A > 0?
Set R-bit to 1
Decrement RQC A
A
Segment waiting for transfer to MCF?
Transfer to segment queue
Transfer contents of RC A to CD A
Set RC A to zero
Increment RQC A

RC A = request counter for bus A

DC A = countdown counter for bus A

RQC A = request queue counter for bus A

MCF = MAC convergence function

= point of introduction of bandwidth balancing procedure (see Section 10.6.4)

10.19 Flowchart of the algorithm used to control the transmission and reception of ments on busA of a dual bus DQDB subnetwork.

slots starts to exceed supply, this has the effect shown in graphica Figure 10.20(a). The access delay variation relates to a heavily loaded work and is the same for both buses. As we can see, the unfairness as the network size increases and/or the bit rate increases.

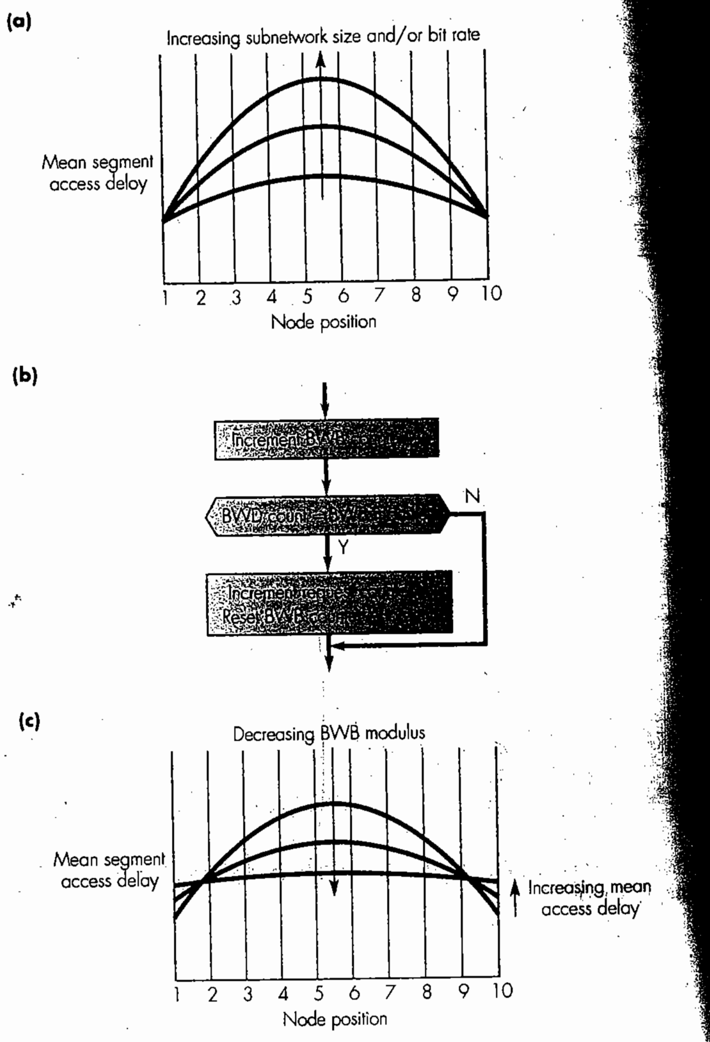


Figure 10.20 Bandwidth balancing: (a) unfairness effect; (b) remedial actions; (c) effect on mean access delay.

To overcome this effect, a modification to the basic access control algorithm known as the **bandwidth balancing mechanism** was introduced. To implement the scheme, a fourth counter called the **bandwidth balancing (BWB) counter** is introduced for each bus. Whenever a segment is transmitted on a bus, the BWB counter for that bus is incremented. Then, whenever the counter reaches a preset limit, the node allows an additional free slot to pass by on this bus by incrementing the corresponding request counter. The BWB counter is then reset to zero and the process repeats. The preset limit is called the **bandwidth balancing modulus.** This operation means that each node, after transmitting a block of segments equal to the BWB modulus, must allow an additional slot to pass by on the related bus. This can be used by the first node lower down the bus which has a queued segment to transmit and a zero countdown counter. The additional processing steps required are shown in Figure 10.20(b) and these are introduced at point *A* in the flowchart shown in Figure 10.19.

The bandwidth balancing mechanism obtains the necessary effect by reducing the utilization of each bus; the smaller the BWB modulus, the larger the bandwidth loss. This is shown in graphical form in Figure 10.20(c), which illustrates the impact of reducing the modulus for a single network type – size and bit rate. As we can see, as the modulus decreases, the unfairness decreases but at the expense of an increase in the mean access delay. In practice, a compromise is made and a value of 8 is normally used. In the worst case, this results in a loss of utilization of 1/(8 + 1) or 11.1%.

10.6.5 Prioritized distributed queuing

Although not always implemented, the basic queued arbitrated access control scheme just described can be extended to support the transmission of prioritized cells/segments. There can be three priority classes and each has a separate set of counters – request counter, countdown counter, and request-queue counter – for each bus. The three priority classes are 0, 1, and 2, with 2 having the highest priority. Cells/segments relating to data-only frames are always allocated class 0, and class 2 is reserved for the transfer of cells relating to network management messages. Class 1 is intended for the transfer of cells containing information that is sensitive to delay or delay variation. Note that bandwidth balancing is not used in this mode.

An example application that may use class 1 cells is for the transfer of compressed video information. As we saw in Section 4.3.1, although the information is generated at a constant rate – determined by the video frame refresh rate – the amount of information associated with each compressed frame varies and depends on the level of movement that has taken place relative to the previous frame. We can deduce that if the isochronous service is used then the (preallocated) bandwidth needs to be that required to transfer a completely new frame at the refresh rate. However, by using the queued arbitrated access method, the amount of information transferred can from one frame to the next. By assigning a higher priority to such segmen the priority control method endeavors to ensure that these are transfer before slots containing other types of LAN traffic.

The general arrangement for controlling access to bus *A* is shown Figure 10.21. For clarity only the request and countdown counters are sho A similar arrangement is used for access to bus *B*. As we can see, there i separate R-bit for each priority class at the head of each slot/cell. These a shown as R0, R1, and R2, where R2 is the highest priority. Assume initiall there are no segments waiting transmission on bus *A* from this node – pa (a). The operation is as follows:

- When an empty slot (B = 0) is repeated at the interface with bus *A*, the access control unit for this bus decrements all three request counters by one.
- When a slot is repeated at the interface with bus *B* – with a priority of 1 for example – the access unit for bus *B* increments only request counters 1 and 0 – RC 1 and RC 0 – and leaves the higher priority counter – RC 2 – unchanged. This means that lower-priority requests do not delay the transmission of higher-priority segments.

Now assume that a segment becomes ready to transmit on bus *A* of priority 1 – part (b). The steps are as follows:

- The current contents of RC 1 are transferred to CDC 1 (and RC 1 is reset to zero) when a slot is repeated at the interface to bus *B* with the corresponding request bit reset to zero.
- When an empty slot is repeated at the interface with bus *A*, request counters RC 0 and RC 2 and countdown counter CDC 1 are decremented.
- If a slot with an R-bit of, say, priority 2 passes, then request counters RC 0 and 2 and countdown counter CDC 1 are incremented.
- The segment is transmitted when CDC 1 becomes zero and an empty slot is received.

Hence incrementing the lower priority countdown counter when a request for a higher priority slot is received, effectively delays the transmission of the lower-priority segment. This means that segments with a higher priority are always transmitted ahead of lower-priority segments.

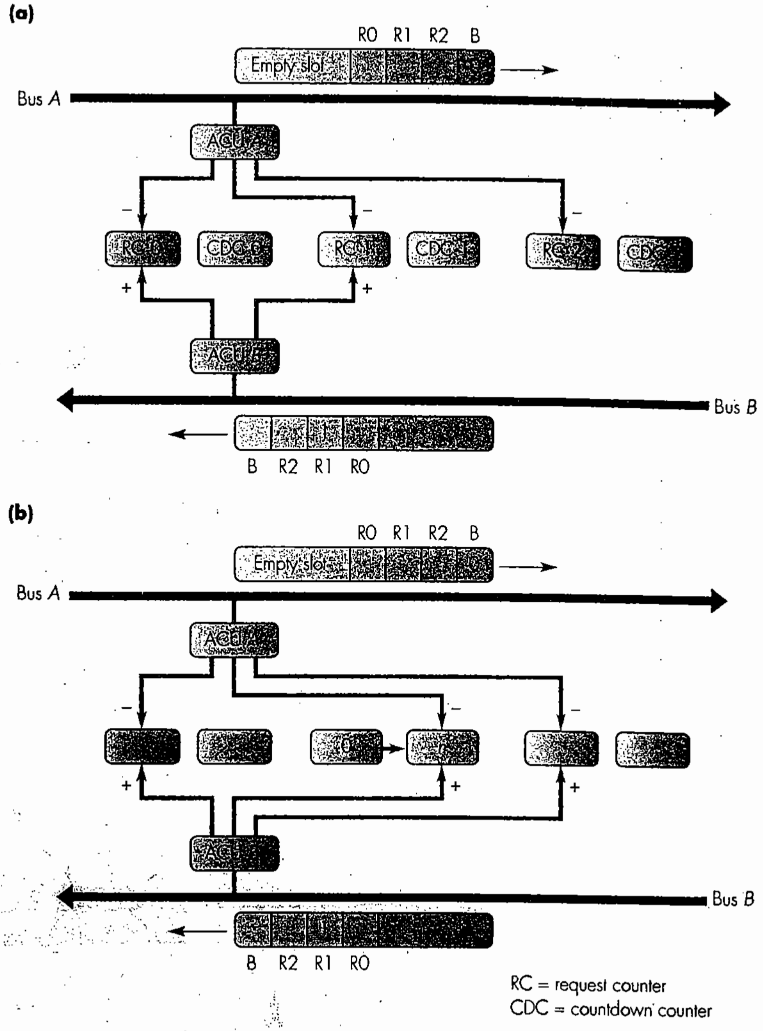


Figure 10.21 Priority access control: (a) no segments waiting; (b) segments queued at priority 1.

10.6.6 Slot and segment formats

As we saw in Section 10.6.2, the 53-octet slot/cell on each bus comprise 5-octet header and a 48-octet payload (contents) field. The structure of header is shown in Figure 10.22(a) and, as we can see, is similar to that us with an ATM network.

The *access control* field contains, in addition to the busy bit and th request bits, a *slot type* bit which indicates whether the slot is to be used f queued arbitrated or prearbitrated (isochronous) data. For slots containi either connection-oriented or isochronous data, the 20-bit *virtual chann identifier* (*VCI*) identifies the logical connection to which the cell conten relate. For LAN (also known as connectionless) data, the VCI is set to all 1 The *payload type* indicates the type of data being carried. It is 00 for all us data – both queued arbitrated and prearbitrated – and the other bit combin tions are used for management information. The *priority* has a default valu of 00 and, for the moment, no other values have been defined. Finally, th *header check sequence* is an 8-bit CRC for error detection purposes.

To transfer connectionless data across the subnetwork – for example, MAC frame between two remote bridges – the submitted frame is first divide

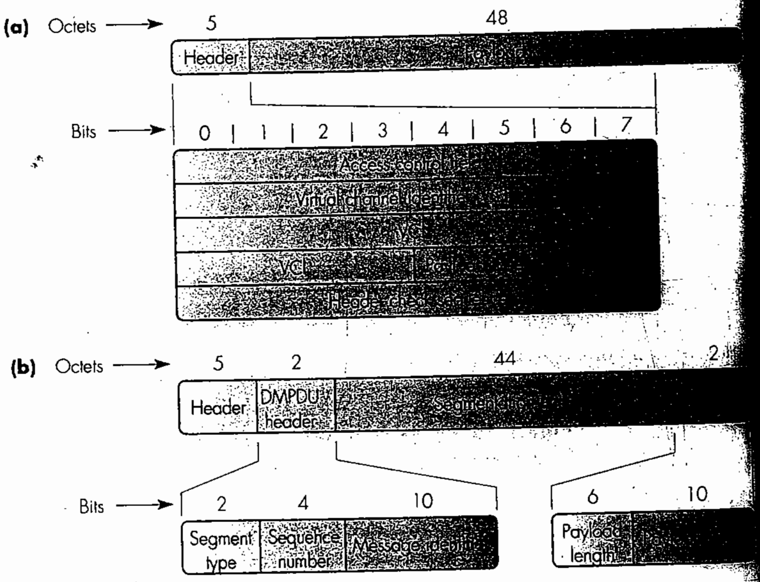


Figure 10.22 Slot and segment formats: (a) slot header; (b) connectionless data segment format.

(segmented) into multiple segments by the MAC convergence protocol in the source access control unit. On receipt, the same protocol at the destination reassembles the received segments back into the original frame. For connection-oriented and isochronous data, the VCI in the cell header is used by the destination to identify those cells which are intended for it. However, for connectionless data, an additional 2-octet header at the start of the 48-octet payload field is used instead. In addition, as Figure 10.22(b) shows, a 2-octet trailer is added at the end of the payload field.

Since segments containing connectionless data are part of a larger (MAC) data frame, they are called **derived MAC protocol data units (DMPDUs)**. The segments relating to a frame – referred to as a message in the standard – are transferred in one of four segment types:

- **single segment message** (SSM): if the frame/message can be carried in a single segment;
- **beginning of message** (BOM): indicates this is the first segment of a multiple segment frame/message;
- **continuation of message** (COM): indicates the contents are between the start and end of a multiple segment frame/message;
- **end of message** (EOM): indicates the last segment.

The *sequence number* and *message identifier* are used together to enable the destination to reassemble the segments relating to a multiple segment message back into its original form.

The *sequence number* is used to detect any missing segments. It is set to zero in the first segment (BOM) and increments for each successive continuation segment (COM) and last segment (EOM). If a missing segment is detected, the remaining segments of the frame/message are discarded.

All segments relating to the same frame are allocated the same *message identifier* (*MID*) by the source access unit. This enables the remaining access units on the bus to identify the segments that relate to the same frame. To ensure these are unique, each access unit is allocated a separate block of identifiers when it is initialized. Clearly, a message identifier is not required in single segment messages and hence it is set to zero.

The trailer comprises two fields: the *payload length* and the *payload CRC*. As we shall see in Section 10.6.7, all submitted frames are padded out to be a multiple of 4 octets. This means that a segment may contain from 4 to 44 octets in multiples of 4 octets. Clearly, not all submitted frames will comprise multiples of 44 octets and hence the payload length indicates the actual number of octets in a single segment message or end of message segment. The payload CRC is a 10-bit CRC and is used to detect transmission errors in the entire 48-octet segment.

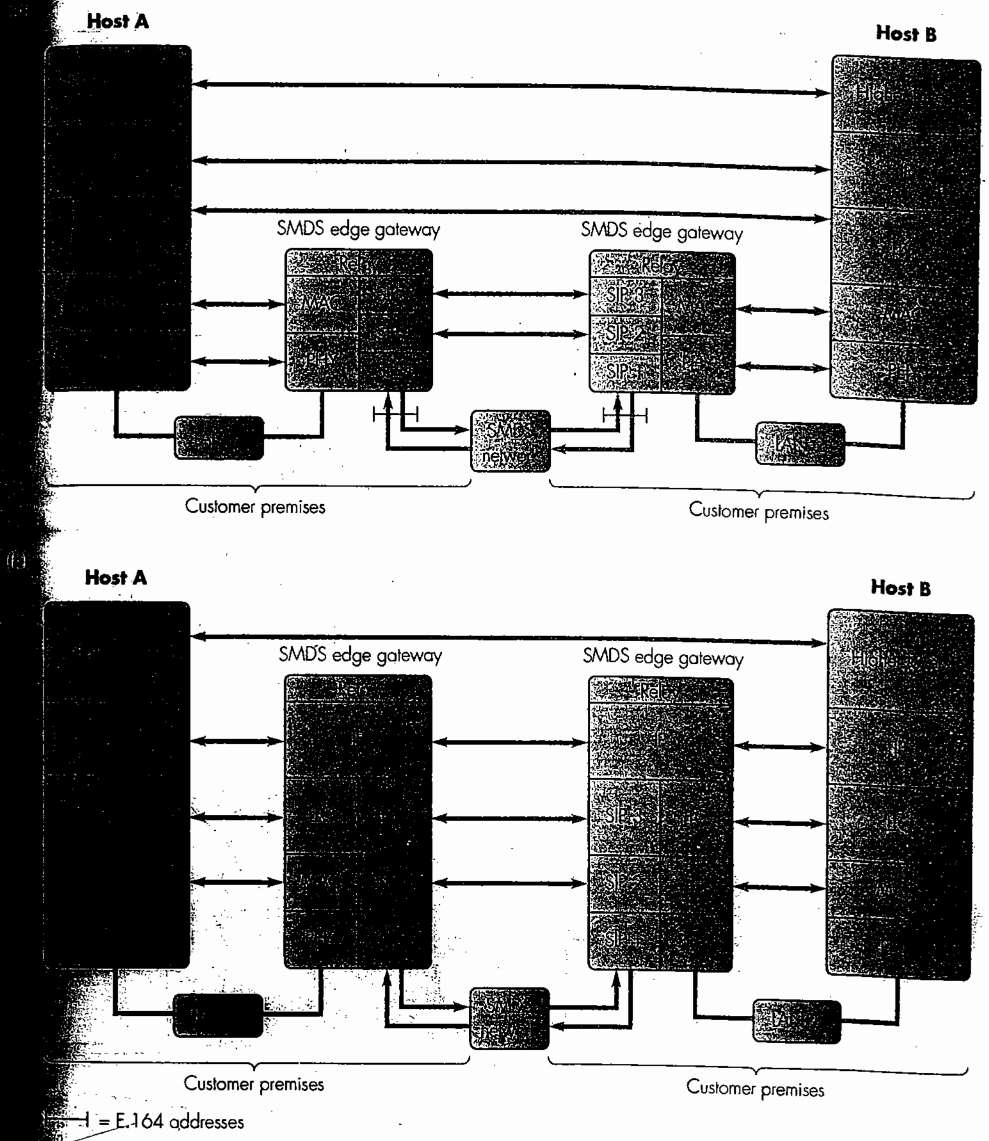
10.6.7 SMDS

In a public network the connectionless data service offered by the MAC co vergence function is known as the switched multimegabit data serv (SMDS) or the **connectionless broadband data service** (**CBDS**) by the ITU In such networks, the various sublayer functions/protocols shown in Figu 10.17 are known as the **SMDS interface protocols** (**SIP**): the MAC conv gence protocol is known as SIP level 3; the queued arbitrated protocol SI level 2; and the physical convergence protocol SIP level 1. A typical interco nection schematic and associated internetworking protocol architecture a shown in Figure 10.23. In part(a) the two LANs are interconnected throug MAC bridges while in part (b) IP routers are used.

Recall from Chapter 8 that the different types of LAN utilize differen header formats, address types, and maximum frame sizes. To accommoda all types of MAC frame, the size of the **SMDS service data unit** can be up 9188 octets. Also, because of the different addressing formats, prior to se menting a submitted frame (SDU), the level 3 SIP first encapsulates th frame between a standard header and trailer. In addition, in order to simpli the buffering operation at the destinations, if necessary, it adds addition padding octets at the tail of the submitted frame so that its length is a mul ple of 4-octets. The resulting message unit – referred to as a packet – known as an initial MAC PDU (IMPDU) – or SIP level 3 PDU – and its forma is shown in Figure 10.24(a).

Thus the SMDS network provides a connectionless service that is tran parent to the customer's internetworking method. To achieve this, on recei of a frame the access gateway – known as an SMDS edge gateway – simp broadcasts the frame over its local DQDB subnetwork using the queue arbitrated access control protocol. In this way, a copy of all submitted frame is received by all the other gateways – access nodes – attached to the sam DQDB subnetwork and, through them, by all the other bridges/routers. decision is made by the latter whether to forward the frame on its LAN simply discard it.

As we can see, the header comprises two or possibly three fields. Th common header contains an 8-bit sequence number – known as the *beg end tag* and used to enable the level 3 SIP to detect missing frames – an specification of the amount of buffer memory required to store the com plete IMPDU. The MAC convergence protocol (MCP) header contain number of subfields relating to the protocol. These include the addresses the source and destination gateways which, in a public network, are 60-b E.164 addresses as defined for use with an ISDN. However, to cater for othe address types these are both 64-bit address fields with the most significant bits identifying the address types used in the remaining 60 bits – for exa ple, 16/48-bit MAC addresses. Other subfields include the number of PA octets present and an indication of whether a CRC is present or not. Th header extension has been included to allow additional subfields to b added in the future.



ure 10.23 SMDS internetworking protocol architectures: (a) bridges; (b) routers.

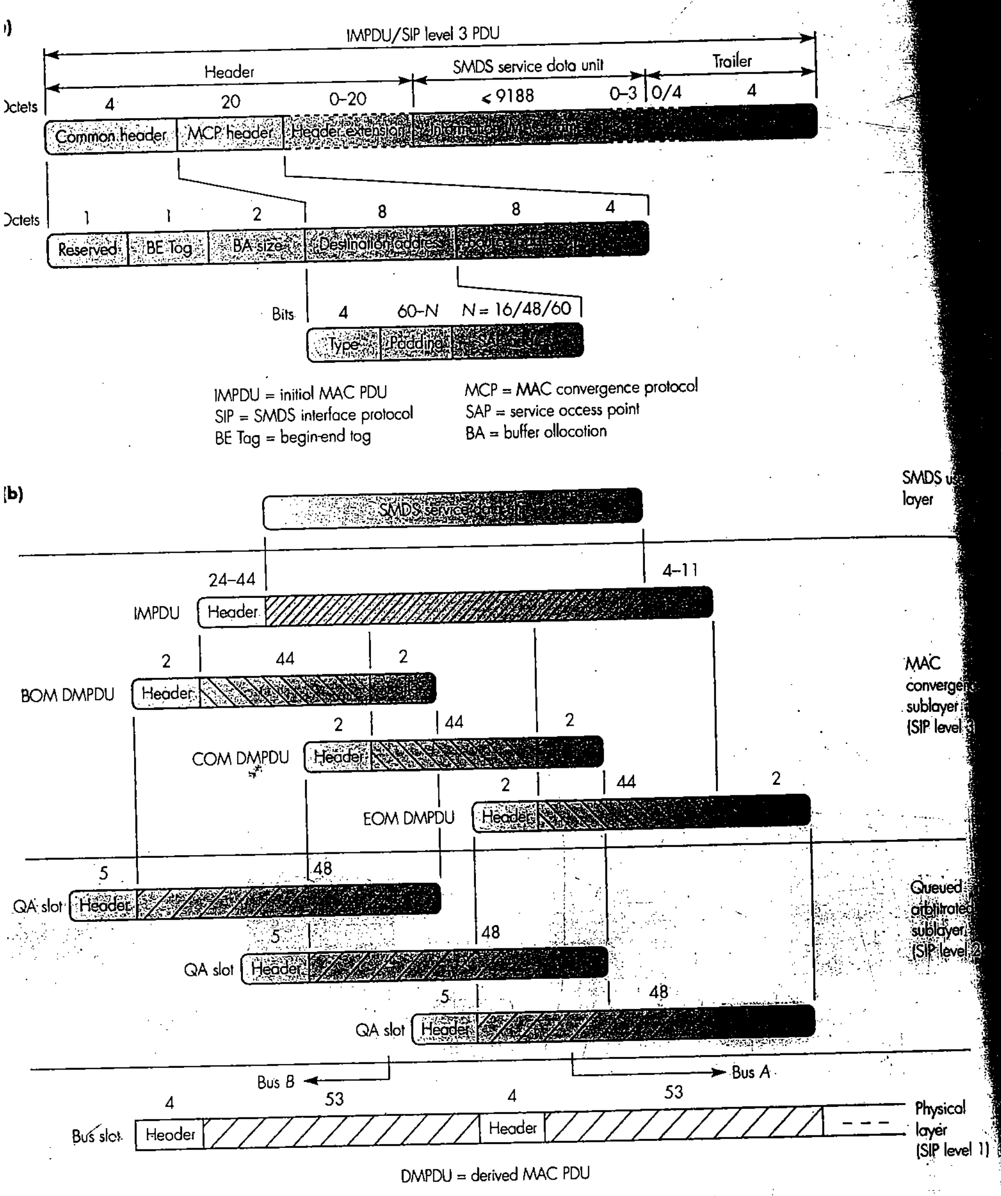

(a)
IMPDU/SIP level 3 PDU
Header
SMDS service data unit
Trailer
Octets
4
20
0–20
≤ 9188
0–3
0/4
4
Common header
MCP header
Header extension
Octets
1
1
2
8
8
4
Reserved
BE Tag
BA size
Destination address
Bits
4
60–N
N = 16/48/60
Type
Padding
SAP
IMPDU = initial MAC PDU
SIP = SMDS interface protocol
BE Tag = begin-end tag
MCP = MAC convergence protocol
SAP = service access point
BA = buffer allocation
(b)
SMDS service data unit
SMDS u
layer
24–44
4–11
IMPDU
Header
2
44
2
BOM DMPDU
Header
COM DMPDU
EOM DMPDU
MAC
converge
sublayer
(SIP level
5
48
QA slot
Header
Queued
arbitrated
sublayer
(SIP level 2
Bus B
Bus A
4
53
Bus slot
Header
Physical
layer
(SIP level 1)
DMPDU = derived MAC PDU

Figure 10.24 Frame transmission overheads: (a) initial MAC PDU format; (b) frame segmentation.

The trailer may include an optional 32-bit CRC which is used for error detection on the complete IMPDU. The common trailer contains the same information as the common header. Hence if the maximum header extension and CRC fields are present and the information field is the maximum 9188 octets, there is an integral number of 210 segments after segmentation.

The steps taken to transfer a submitted MAC frame across the SMDS network, together with the overheads associated with each sublayer function, are summarized in Figure 10.24(b). The submitted SMDS service data unit is first encapsulated by the MAC convergence protocol to form an IMPDU. It then segments this into a number of DMPDUs, each with the corresponding header and trailer. The resulting 48-octet segments are passed to the queued arbitrated function which adds the appropriate 5-octet header to them. Finally these are passed to the common function which initiates their transmission via the physical layer convergence sublayer. We can best quantify the overheads associated with each of these functions using an example.

Example 10.3

A 510-octet MAC frame is to be transferred across an SMDS/DQDB subnetwork. Stating clearly any assumptions you make, derive the number of queued arbitrated slots that are required to carry out the transfer and hence the total number of overhead octets involved.

Answer:

With reference to Figure 10.24(b):

- MAC convergence protocol:
  - add 2 octets to make the frame 512 octets which is an integral multiple of 4 octets
  - assuming a header extension and CRC are not used, a 24-octet header and a 4-octet trailer are added to create a 540-octet IMPDU
  - total overheads = 2 + 24 + 4 = 30 octets
  - after segmentation the IMPDU requires 13 DMPDUs, 12 containing a full complement of 44 octets and one with 12 octets
  - total overheads are 4 for each of the 13 DMPDUs = 52, plus 32 for the part full EOM DMPDU
- Queued arbitrated (QA) sublayer:
  - a further 5 octets are added to each 48-octet DMPDU to give 13 QA slots
  - total overhead = 5 × 13 = 65 octets.

ontinued

- Physical layer:
  - a further 4-octet header is added to each QA slot
  - total overheads = 4 × 13 = 52 octets.
- In summary:
  - QA slots required = 13
  - total overheads = 30 + 52 + 32 + 65 + 52 = 231 octets.

10.7 Wide area ATM networks

As we showed earlier in Figure 10.15(c), a number of network providers hav created wide area ATM networks by interconnecting multiple MAN subnet works together. As we indicated, each MAN subnetwork can be implemente using either DQDB or conventional ATM switches. Each MAN subnetwork is connected to a MAN switching system (MSS) and these in turn are intercon nected by means of high bit rate lines. For the MSSs to perform thei routing/switching function, the E.164 addresses in the header of eac IMPDU are networkwide addresses which identify the MSS and subnetwork t which each access gateway is attached. In addition, group addressing is sup ported and, when the access node receives an IMPDU with a preassigne group address in its destination address field, a copy of the IMPDU is sent t all members of the group.

The interconnected LANs in the two protocol architectures we showed i Figure 10.23 were assumed to be frame-based LANs. However, a simila approach can be used to interconnect ATM LANs. Figure 10.25(a) shows typi cal arrangement, together with the associated protocol architecture. In th example we assume the connectionless service associated with the ATM LA is provided at the IP layer. Hence, as we showed in Figure 10.14 an described in the accompanying text, a connectionless server (CLS) is use However a similar approach is used when the connectionless service is bein provided at the MAC sublayer with a LAN emulation server (LES).

As we can see in Figure 10.25(a), an ATM gateway is located at each cus tomer site, one port of which is connected to the site ATM LAN and th other port to an ATM MAN. Typically, there are a number of gateways (sites) connected to the MAN in a particular area and this provides a local gathering and distribution function. As before, an MSS is connected to each MAN and the MSSs are interconnected together to form a wide area switched network. The service provided is the SMDS/CBDS and the ATM LAN gateway is the SMDS edge gateway.

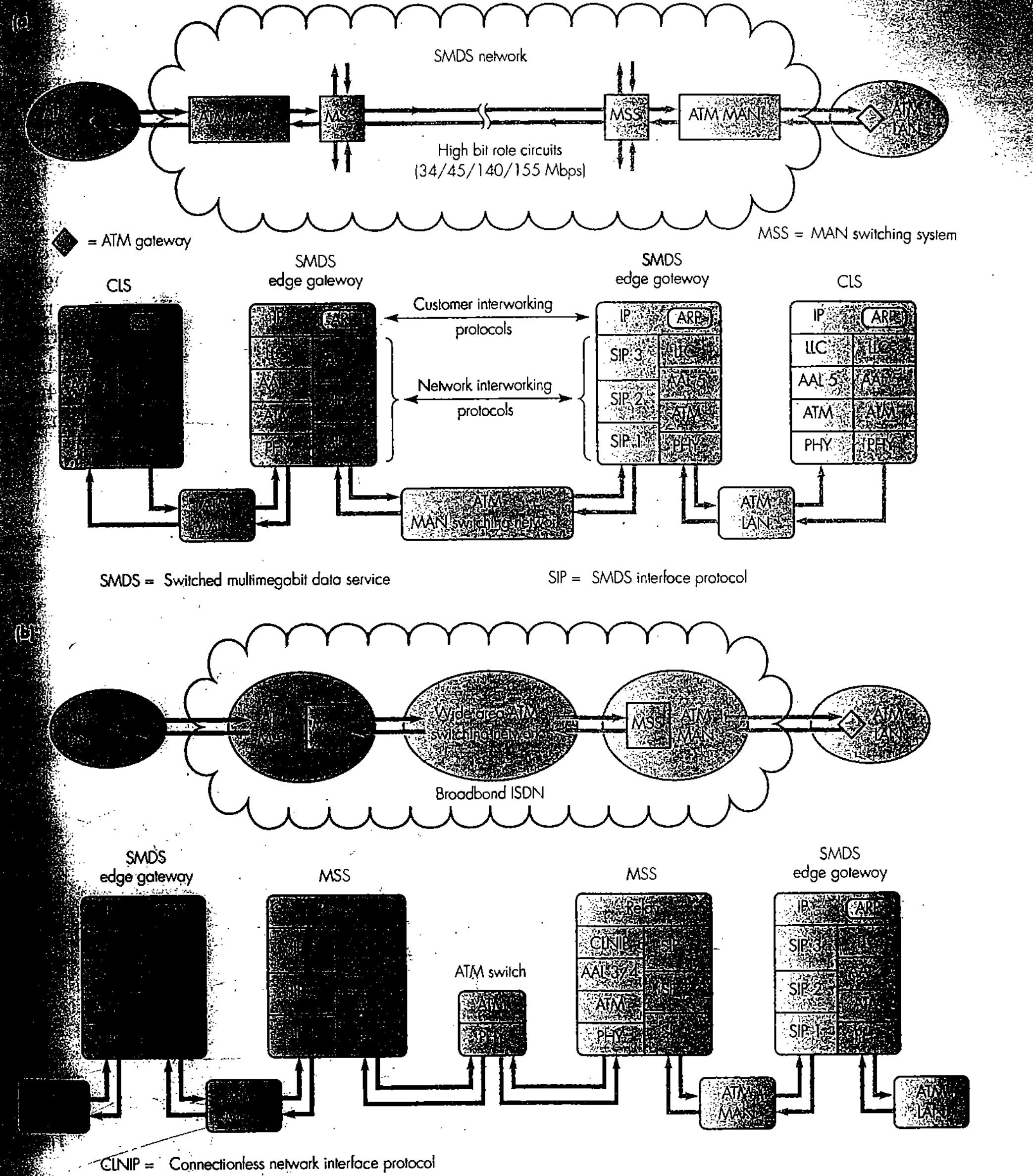

(a)
SMDS network
ATM MAN
MSS
MSS
ATM MAN
ATM LAN
High bit rate circuits
(34/45/140/155 Mbps)
= ATM gateway
MSS = MAN switching system
CLS
SMDS
edge gateway
SMDS
edge gateway
CLS
Customer interworking
protocols
Network interworking
protocols
IP
ARP
AAL 5
ATM
PHY
IP
ARP
SIP 3
SIP 2
SIP 1
LLC
AAL 5
ATM
PHY
IP
ARP
LLC
LLC
AAL 5
AAL 5
ATM
ATM
PHY
PHY
ATM
LAN
ATM
MAN switching network
ATM
LAN
SMDS = Switched multimegabit data service
SIP = SMDS interface protocol
(b)
Wide area ATM
switching network
MSS
ATM
MAN
ATM
LAN
Broadband ISDN
SMDS
edge gateway
MSS
ATM switch
MSS
SMDS
edge gateway
ATM
PHY
Relay
CLNIP
AAL 3/4
ATM
PHY
SIP
IP
ARP
SIP 3
SIP 2
SIP 1
ATM
MAN
ATM
LAN
CLNIP = Connectionless network interface protocol

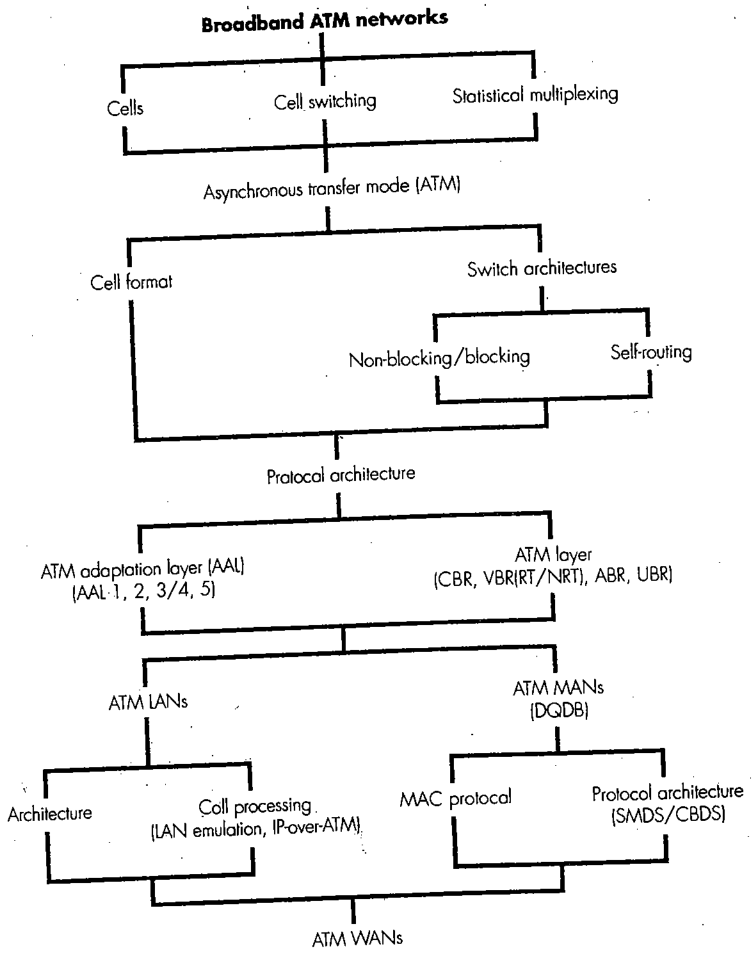
Figure 10.25 Connectionless working over wide area ATM networks: (a) ATM MAN switching network; (b) broadband ISDN.

On receipt of an IP datagram/packet with a destination address th cates a different site network (netid), the CLS within the ATM LAN rel packet to the ATM LAN port of the SMDS edge gateway. The packe MAC frame if LAN emulation is being used – is first encapsulated into dardized frame format by the SIP level 3 protocol and then relayed acro SMDS network to the appropriate destination gateway. From there relayed first to the site CLS and from there to the destination station.

In some networks, the MSSs are interconnected by an ATM switching work rather than point-to-point high bit rate lines. This type of netwo together with the protocol architecture associated with it, is shown in Fig 10.25(b). As we can see, this is similar to part (a) except in this case an inter diate ATM switching network is present. The **connectionless network interf protocol (CLNIP)** provides a similar service to the SIP level 3 protocol and format of each CLNIP PDU is similar to that given earlier in Figure 10.24(The main difference is that there is no common header or trailer with CLNIP PDU since this is provided by the AAL in an ATM network.

Summary

A summary of the topics discussed in this chapter is given in Figure 10.26.


Broadband ATM networks
Cells
Cell switching
Statistical multiplexing
Asynchronous transfer mode (ATM)
Cell format
Switch architectures
Non-blocking/blocking
Self-routing
Protocol architecture
ATM adaptation layer (AAL) (AAL 1, 2, 3/4, 5)
ATM layer (CBR, VBR(RT/NRT), ABR, UBR)
ATM LANs
ATM MANs (DQDB)
Architecture
Cell processing (LAN emulation, IP-over-ATM)
MAC protocol
Protocol architecture (SMDS/CBDS)
ATM WANs

gure 10.26 Broadband ATM networks, chapter summary.

Exercises

n 10.1

xplain the origin/meaning of the following erms relating to B-ISDN networks:
- i) fixed-size cells,
- ii) statistical multiplexing,
- iii) cell switching,
- iv) asynchronous transfer mode.

iscuss the reasoning behind the choice of a onnection-oriented mode of operation for B-SDN networks and a 53-byte cell size.

xplain why the development of B-ISDN has een postponed. Identify those networks and etworking equipment of that use ATM.

10.2

ith the aid of the cell switching schematic own in Figure 10.1, explain
-) how the header of each cell can be relatively short,
- i) how cells are routed through an ATM switch,
- ii) the difference between VP routing and VC routing. Give an example of each routing type.

ith the aid of the ATM format shown in igure 10.2, explain the use of
-) the payload type field,
- i) the cell loss priority bit,
- ii) the header checksum.

10.3

ith the aid of the switch architecture shown Figure 10.3(a), explain the principle of peration of an ATM switch. Include in your planation the role of the input and output ntrollers, the switching matrix and the itch control processor. Also explain why cell affers are required in the input and output ntrollers.

10.7 With the aid of the diagrams shown in Figure 10.3(b) and (c), explain the principle of operation of the following ATM switch matrix types:
- (i) time division bus,
- (ii) fully-connected.

Assuming they are to operate in a nonblocking mode, estimate the maximum number of ports associated with each matrix type and what determines this.

10.8 Explain why large fabrics consist of multiple switching stages each made up of a number of smaller switching elements interconnected in a regular matrix.

10.9 In the context of a multistage delta switching matrix comprising a number of switching elements, if M is the number of input ports and is the number of inputs per switching element, derive expressions for the number of switching elements per stage, X, and the number of stages, Y.

10.10 With the aid of the 8-port delta switching matrix shown in Figure 10.4, explain how a cell is routed through the matrix and how blocking can occur.

10.11 An example of a switching matrix that avoids blocking is the Batcher–Banyan switch matrix shown in Figure 10.5. Explain the principle of operation of such switches and how blocking is avoided.

10.12 Using the Batcher–Banyan switch matrix shown in Figure 10.5, identify the paths through the matrix followed by the following set of cells. Assume they arrive simultaneously at the eight input ports and the routing tags given start at port 1:

111, 100, 011, 000, 101, 001, 110, 010

Assume now that the routing tag of the call arriving at port 4 is 111 instead of 000. Determine the effect of this and how it is overcome in practice.

Section 10.4

10.13 With the aid of the protocol architecture shown in Figure 10.6, explain
- (i) the role of the signaling control point and the network management stations,
- (ii) the meaning of the C, U, and M planes,
- (iii) the role of the AAL and ATM layer protocols.

10.14 With the aid of the diagram shown in Figure 10.7, state the application domain of each of the four AAL service classes. Also explain the role of the CS and SAR sublayers in implementing these services. Include the use of the service access point.

10.15 With the aid of the two SAR PDU types shown in Figure 10.8, explain the use of the following fields:
- (i) SN,
- (ii) SNP,
- (iii) IT.

10.16 Explain the difference between the AAL3/4 and AAL5 layers and why the latter was developed. Hence, using the PDU formats of the CS and SAR sublayers shown in Figure 10.9, derive the number of protocol overhead octets/bytes required to send an AAL service data unit of 1024 bytes using
- (i) AAL3/4,
- (ii) AAL5.

10.17 Explain the role of the ATM layer including the meaning of the following service classes:
- (i) CBR,
- (ii) VBR/RT,
- (iii) VBR/NRT,
- (iv) ABR,
- (v) UBR.

10.18 Explain the meaning of the terms "traffic descriptor" as applied to a call and the use of the following parameters associated with it:
- (i) PCR,
- (ii) SCR,
- (iii) MCR,
- (iv) CDVT,
- (v) CLR,
- (vi) CTD,
- (vii) CDV.

10.19 In order to ensure the agreed operational parameters for a call are being adhered to, the network uses the generic cell rate algorithm. With the aid of the four example cell interarrival times shown in Figure 10.10, explain the principle of operation of the algorithm.

Section 10.5

10.20 In relation to the schematic diagram shown in Figure 10.11 of an ATM LAN explain the role of the following:
- (i) an RCU,
- (ii) the broadcast server,
- (iii) the signaling control point and how stations communicate with it,
- (iv) the LAN emulation server.

10.21 With the aid of the connectionless protocol architecture shown in Figure 10.14, outline the steps followed to enable:
- (i) a cell-based client to access a cell-based server
- (ii) a PC/workstation attached to a legacy LAN to access a server that is attached to a different legacy LAN (of the same type) both of which are connected to the ATM backbone by bridges
- (iii) a PC/workstation that is attached to a legacy LAN to access a server that is attached directly to the ATM backbone. Assume the legacy LAN is connected to the ATM backbone by a router.

Section 10.6

10.22 With the aid of the three DQDB MAN architectures shown in figure 10.15, explain the meaning/use of
- (i) dual contradirectional buses,
- (ii) access node/customer network interface unit,
- (iii) open bus and dual bus topologies,
- (iv) isochronous gateway,
- (v) MAN switching system,
- (vii) SMDS.

10.23 With the aid of the DQDB architecture shown in Figure 10.16(a) and (b), state the differences between an open bus and a loop us architecture. Explain the principle of peration of each type.

Vith the aid of the DQDB architecture hown in Figure 10.16(c). describe how a ooped bus architecture can continue working in the presence of
(i) a communications link failure and
(ii) a node failure.

With the aid of the DQDB protocol architecture shown in Figure 10.17, explain the meaning/use of
(i) queued arbitrated and MAC convergence functions,
(ii) prearbitrated and isochronous conversion function,
(iii) common functions,
(iv) physical layer convergence function.

With the of a diagrams in Figure 10.18 and the flowchart in Figure 10.19, explain the principle of operation of the queued-packet distributed-switch (QPSX) access control method. Include the meaning/use of:
(i) the busy and request bits in the header of each slot,
(ii) the request and countdown counters,
(iii) the request-queue counter.

With the aid of Figure 10.20(a), explain why bandwidth balancing is necessary with the QPSX access control method. Hence explain how the remedial actions shown in Figure 10.20(b) overcome the unfairness arising with the basic scheme.

With the aid of the access control schematic shown in Figure 10.21, explain the principle of operation of the distributed priority queuing mechanism used with multimedia applications. Include the role of the various counters and the effect on their contents of the request and busy bits when
(i) no segments are queued for transmission, and
(ii) a segment is queued at a specified priority level.

Define the meaning of the terms "DQDB slot" and "DQDB segment". Hence with the aid of the formats shown in Figure explain the role of the slot and se headers.

10.30 Assuming a frame of data is to be transf over a DQDB subnetwork and the fr requires multiple slots, with the aid of segment format shown in Figure 10.2 explain the function of each field in DMPDU header and trailer. Describe the steps followed by the receiver wh reassembling the frame.

10.31 Define the service offered by an SMDS work. Hence with the aid of the alterna protocol architectures shown in Fig 10.23, explain the function of the SIP lev and SIP level 2 protocols when the SM edge gateway is
(i) a bridge and
(ii) a router.

10.32 Define the format of the source and desti tion addresses used in the header of an in MAC PDU and why the size of an SMDS vice data unit can be up to octets/bytes.

10.33 A 100-byte MAC frame is to be transfer access a DQDB SMDS network. Using transmission overheads identified in Fig 10.24, derive:
(i) the number of queued arbitrated s that are required to carry out the tra fer of the frame,
(ii) the total number of overhead by State any assumptions you make.

Section 10.7

10.34 With the aid of the ATM MAN switching work and associated protocol architect shown in Figure 10.25(a), explain
(i) the role of the MAN switching syst including the addressing mechanis associated with it,
(ii) how an IP datagram/packet is rout across the total network.

10.35 Identify and explain the differences when a ATM switch is used within the SMDS networ Include the role of the CLNIP protocol i the MSS.

11

Entertainment networks and high-speed modems

11.1 Introduction

As we saw in Section 1.4.3, entertainment applications include movie/video-on-demand, near video-on-demand, broadcast television, and interactive television. Also, as we saw in Section 4.3.4, the bit rate requirements for the (compressed) audio and video associated with these applications are determined by the bandwidth of the transmission channel to be used. Typical bit rates are:

VCR-quality video with sound:	1.5 Mbps (MPEG-1)
Broadcast-quality video with sound:	4/6/8 Mbps (MPEG-2, Main)
Studio-quality television with sound:	9/15/18 Mbps (MPEG-2, Main)
High-definition television with sound:	60/80 Mbps (MPEG-2, High)

overheads associated with the other blocks in the channel interface – the R (204, 188) coder and a rate 3/4 convolutional coder – reduce this figure t about 16 Mbps (24 × 184/272). This can be used to carry, say, four 4Mbp digital TV programs in each 8 MHz broadcast channel.

11.4.3 Interactive services

The interactive services available with terrestrial broadcast networks are sim lar to those we identified in relation to satellite networks. In genera however, the amount of transmission bandwidth/number of channels available with terrestrial networks is much less than with satellite networks. Hence the range of services available is only a subset of those listed in Section 11.3.3

11.5 High-speed PSTN access technologies

As we saw in Figure 1.1(c), the access network of a PSTN, in addition to supporting the plain old telephone service (POTS) for which it was designed, now supports a number of additional services. For example, a range of low bi rate data applications such as fax are supported by means of low bit rate – less than 56 kbps – modems. Also, as we saw in Section 7.2.3, twisted-pair lines in the PSTN access network are used as the access lines for an ISDN. Bit rates of between 144 kbps (basic rate) and 1.544/2.048 Mbps (primary rate) over several miles/kilometers are obtained using baseband transmission. The access line is then known as a digital subscriber line (DSL). In the case of a basic rate line, this is called an **ISDN DSL** (**IDSL**) and, in the case of a primary rate line, a **high-speed DSL** (**HDSL**). An IDSL uses a single pair and an HDSL two pairs. In addition there is a simpler version of HDSL which operates over a single pair. This is known as **single-pair DSL** (**SDSL**) and bit rates of up to 1.544/2.048 Mbps are supported depending on line length.

Both the basic rate and the primary rate lines of an ISDN are symmetric that is, they operate with an equal bit rate in both directions. However, as we saw earlier in Section 11.2.1, with most interactive applications the information flow is asymmetric and involves a low bit rate channel from the subscriber for interaction purposes and a high bit rate channel in the downstream direction for the return of the requested information. Asymmetric ratios of from 10:1 to in excess of 100:1 are common. In order to exploit this many cable operating companies have introduced cable modems that support various applications of this type.

In addition, many **telecommunication operating companies** (**telcos**) have introduced additional types of DSL technologies to meet these same requirements over the twisted-pair lines used in most PSTN access networks. Unlike the various DSL technologies associated with an ISDN, these have been designed to enable the signals associated with the existing telephony services to coexist with those associated with the newer high-speed interactive services on the same twisted-pair line. Two types are used: the first known as **asymmetric DSL** (**ADSL**) and the second **very-high-speed DSL** (**VDSL**). In the case of ADSL, the high-speed asymmetric channel is designed to coexist with the existing analog telephony service. In the case of VDSL, the high-speed channel, in addition to operating at a higher bit rate than that of an ADSL, can operate in either an asymmetric or symmetric mode and is designed to coexist with either analog telephony or basic-rate ISDN services. In this section we discuss both these technologies.

11.5.1 ADSL

The standard relating to ADSL was produced by the ANSI in 1995 and is defined in **T1.413**. It was defined originally to meet the requirements of broadcast-quality video-on-demand (VOD). Hence the standard allows for a bit rate of up to 8 Mbps in the downstream direction – that is, from the local exchange/end office (LE/EO) to the customer premises – and up to 1 Mbps in the upstream direction. In practice, however, high-speed access to the Internet proved to be more popular than VOD. As a result, since the bit rate and QOS requirements associated with Internet applications are less than those of VOD, a variant of the original ADSL standard known as **ADSL-Lite** (or sometimes **G-Lite**) has been defined. This has been developed within the ITU and is defined in standard **G.992.2**. It provides a downstream bit rate of up to 1.5 Mbps and an upstream bit rate of up to 384 kbps. As with ADSL, the actual bit rates achievable depend on the length and quality of the line. Nevertheless, the lower bit rates associated with ADSL-Lite means that it can be used over longer distances and with poorer-quality lines than ADSL. In addition, as we shall see, ADSL-Lite can be used with a passive network termination at the customer premises rather than the active termination that is required with ADSL.

Access network architectures

As we saw in Section 7.2.1, in the earliest PSTN access networks the transmission lines were made up entirely of interconnected sections of unshielded cable containing multiple twisted-pair wires. Over a period of time, however, sections of these cables have been replaced with optical fiber cable. The architecture of a typical PSTN access network is as shown in Figure 11.22.

As we can see, the amount of twisted-pair cable used varies from zero with **fiber-to-the-home** (**FTTH**) or **fiber-to-the-building** (**FTTB**), short (less than 300 yards/meters) drop cables with **fiber-to-the-kerb/curb** (**FTTK/C**), one or two cable sections with **fiber-to-the-cabinet** (**FTTCab**), and all twisted-pair for direct-to-building cable runs of up to 2.5 miles (4km). Hence, since the objective of ADSL is to provide high-speed interactive services over the twisted-pair portion of the access line, ADSL is designed on the assumption that the maximum length of twisted-pair cable is less than 2.5 miles (4 km).

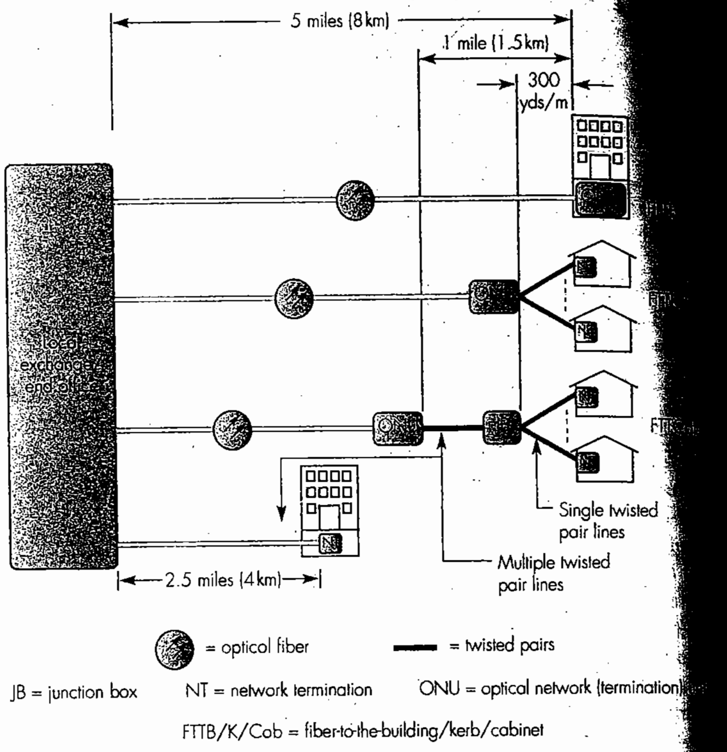

5 miles (8km)
1 mile (1.5km)
300 yds/m
Local exchange/end office
FTTB
FTTK
FTTCab
NT
JB
ONU
Single twisted pair lines
Multiple twisted pair lines
2.5 miles (4km)
= optical fiber
= twisted pairs
JB = junction box
NT = network termination
ONU = optical network (termination) unit
FTTB/K/Cob = fiber-to-the-building/kerb/cabinet

Figure 11.22 Typical modern access network architecture.

Connection alternatives

As we saw in Section 2.5.1, the lower frequency band up to 4 kHz of the
width available with a (single) twisted-pair line is used for analog telep
(POTS). So in order for the two ADSL/ADSL-Lite signals to coexist wi
POTS signal on the same line, modulated transmission must be used to
the signals away from the lower frequency band. In the case of ADSL, th
signals are transmitted in the frequency band from 25 kHz throug
1.1 MHz and, for ADSL-Lite, the upper frequency is limited to 500 kHz.
components that are used to deliver both services over the same access li
are shown in Figure 11.23. The arrangement shown in part (a) relates to a ty
ical ADSL installation and that in part (b) an ADSL-Lite installation. In bo
cases it is assumed that the access line is all twisted-pair and this terminates i
the LE/EO. With the various fiber access alternatives, the same line termina
tion arrangement is located in the **optical network (termination) unit (ONU)**.

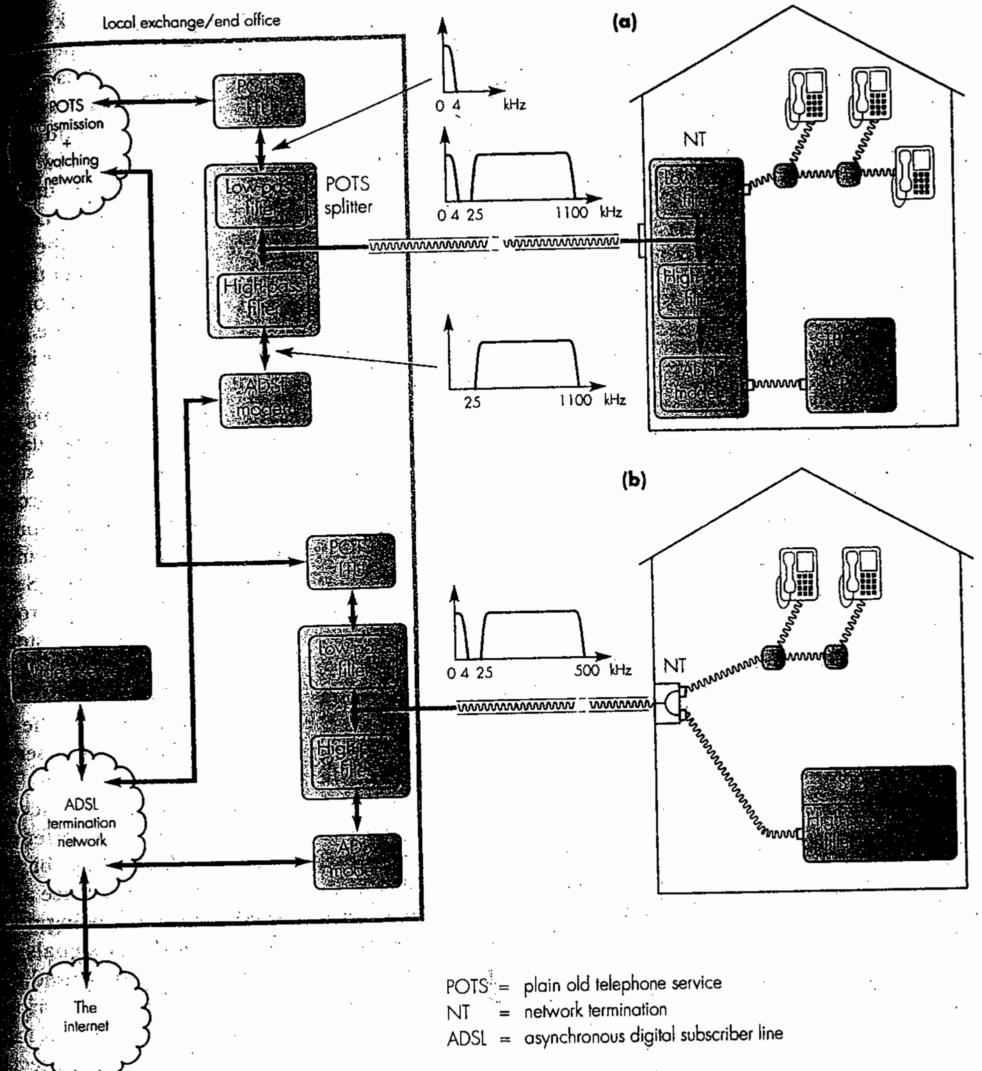

Local exchange/end office
(a)
(b)
POTS transmission + switching network
POTS LTU
Low-pass filter
High-pass filter
POTS splitter
ADSL modem
ADSL termination network
The internet
0 4 kHz
0 4 25 1100 kHz
25 1100 kHz
0 4 25 500 kHz
NT
POTS = plain old telephone service
NT = network termination
ADSL = asynchronous digital subscriber line

Figure 11.23 Connection alternatives: (a) ADSL with active NT; (b) ADSL-Lite with passive NT.

33 With the aid of the diagrams relating to COFDM in Figures 11.20 and 11.21, explain the implications of:
- (i) using orthogonal carriers on the spacing between adjacent carriers,
- (ii) adding a guard interval between symbols,
- (iii) the relationship between the guard interval and the symbol period.

4 Assuming a guard interval of 50 μs and a usable channel bandwidth of 7 MHz, derive the number of carriers that would be used.

5 Define the origin of 2K and 8K in the 2K/8K specifications for digital video broadcasting-terrestrial (DVB-T).

With the aid of the 2K/8K frame formats shown in Figure 11.21, explain the meaning/use of the following relating to receiver synchronization:
- (i) a frame,
- (ii) continual pilots,
- (iii) scattered pilots,
- (iv) TPS pilots.

With reasons, estimate the usable bit rate that is available with an 8 MHz broadcast channel assuming the 2K specification, and 16-QAM modulation per carrier. Allow for loss of bit rate caused by the guard interval and the channel interface overheads quantified in Example 11.1. State the number of digital TV programs the channel will support.

on 11.5

State the meaning, bit rate, and use of the following PSTN high-speed access technologies:
- (i) IDSL,
- (ii) HDSL,
- (iii) SDSL,
- (iv) ADSL-Lite,
- (v) ADSL,
- (vi) VDSL.

11.39 With the aid of the schematic diagram of typical modern access network architecture shown in Figure 11.22, state the length of twisted-pair cable with each of the following:
- (i) FTTH,
- (ii) FTTK/C,
- (iii) FTTCab,
- (iv) direct-to-building.

Hence state the maximum length of twisted pair cable that can be used with ADSL.

11.40 With the aid of the access network architecture shown in Figure 11.23(a), describe how the POTS and high-speed interactive services are provided with an ADSL installation. Include in your description the role of the low-pass and high-pass filters in the POTS splitter and the ADSL modem.

11.41 With the aid of the access network architecture shown in Figure 11.23(b), describe how the POTS and high-speed interactive services are provided with an ADSL-Lite installation. Include in your description the advantages of using a passive network termination (NT) and one of the potential drawbacks of this.

11.42 In relation to the discrete multitone (DMT) modulation method used with ADSL modems, state:
- (i) the number of carriers per symbol that are used,
- (ii) the reason why multilevel modulation is not used with all carriers,
- (iii) how duplex transmission is obtained.

11.43 State the following for a VDSL installation:
- (i) the type of access network and the location of the VDSL modem,
- (ii) typical bit rates,
- (iii) how duplex transmission is obtained.

12

Transport protocols

12.1 Introduction

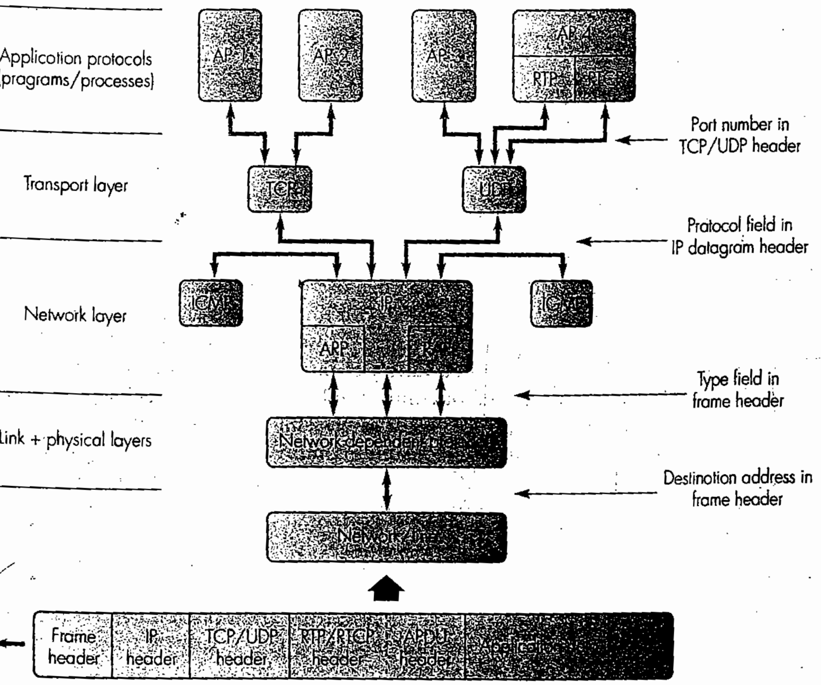
As we saw earlier in Section 5.2, although the range of multimedia applications (and different types of network used to support them) are many and varied, the protocol suites associated with the different application/network combinations have a common structure. As we saw in Section 1.5.3, the different types of network operate in a variety of modes – circuit-switched or packet-switched, connection-oriented or connectionless – and hence each type of network has a different set of protocols for interfacing to it. Above the network-layer protocol, however, all protocol suites comprise one or more application protocols and a number of what are called application-support protocols.

For example, in order to mask the application protocols from the services provided by the different types of network protocols, all protocol suites have one or more transport protocols. These provide the application protocols with a network-independent information interchange service and, in the case of the TCP/IP suite, they are the **transmission control protocol (TCP)** and the **user datagram protocol** (UDP). TCP provides a connection-oriented (reliable) service and UDP a connectionless (best-effort) service. Normally, both protocols are present in the suite and the choice of protocol used is determined by the requirements of the application. In addition, as we saw in

Figure 5.6, when the application involves the transfer of streams and/or video in real time, the timing information required by the re synchronize the incoming streams is provided by the **real-time transp tocol (RTP)** and its associated **real-time transport control protocol** In this chapter we describe the operation of these four protocols and vices they provide to application protocols.

12.2 TCP/IP protocol suite

Before describing the various protocols, it will be helpful to illustrate the tion of each protocol relative to the others in the TCP/IP suite. This is in Figure 12.1. Normally, the IP protocols and network-dependent pro below them are all part of the operating system kernel with the various cation protocols implemented as separate programs/processes. Th transport protocols, TCP and UDP, are then implemented to run within the operating system kernel, as separate programs/processes, o library package linked to the application program.



.2.1 **TCP/IP protocol suite and interlayer address selectors.**

With most networked applications, the client–server paradigm is used. The client application protocol/program runs in one computer – typically a PC or workstation – and this communicates with a similar (peer) application program that runs (normally continuously) in a remote server computer. Examples of applications of this type are file transfers and the messages associated with electronic mail, both of which require a reliable service; that is, the transferred information should be free of transmission errors and the messages delivered in the same sequence that they were submitted. Hence applications of this type use the reliable service provided by TCP.

Thus the role of TCP is to convert the best-effort service provided by IP into a reliable service. For other applications, a simple best-effort service is acceptable and hence they use UDP as the transport protocol. Examples of applications of this type are interpersonal applications that involve the transfer of streams of (compressed) audio and/or video in real time. Clearly, since new information is being received and output continuously, it is inappropriate to request blocks that are received with errors to be retransmitted. Also, it is for applications of this type that RTP and RTCP are used. Other applications that use UDP are application protocols such as HTTP and SNMP, both of which involve a single request-response message exchange.

As we saw in Figure 9.1 and its accompanying text, all message blocks – protocol data units (PDUs) – relating to the protocols that use the services of the IP layer are transferred in an IP datagram. Hence, as we can deduce from Figure 12.1, since there are a number of protocols that use the services of IP – TCP, UDP, ICMP, and IGMP – it is necessary for IP to have some means of identifying the protocol to which the contents of the datagram relate. As we saw in Section 9.2, this is the role of the *protocol* field in each IP datagram header. Similarly, since a number of different application protocols may use the services of both TCP and UDP, it is also necessary for both these protocols to have a field in their respective PDU header that identifies the application protocol to which the PDU contents relate. As we shall see, this is the role of the source and destination *port numbers* that are present in the header of the PDUs of both protocols. In addition, since a server application receives requests from multiple clients, in order for the server to send the responses to the correct clients, both the source port number and the source IP address from the IP datagram header are sent to the application protocol with the TCP/UDP contents.

In general, within the client host, the port number of the source application protocol has only local significance and a new port number is allocated for each new transfer request. Normally, therefore, client port numbers are called **ephemeral ports** as they are short-lived. The port numbers of the peer application protocol in the server application protocols are fixed and are known as **well-known port numbers.** Their allocation is managed by IANA and they are in the range 1 through 1023. For example, the well-known port number of the server-side of the file transfer (application) protocol (FTP) is 21. Normally, ephemeral port numbers are allocated in the range 1024 through to 5000.

As we can see in Figure 12.2, all the protocols in both the application an transport layers communicate directly with a similar peer protocol in th remote host computer (end system). The protocols in both these layers ar said, therefore, to communicate on an end-to-end basis. In contrast, the I protocols present in each of the two communicating hosts are networ interface protocols. These, together with the IP in each intermediate gat way/router involved, carry out the transfer of the datagram across th internetwork. The IP protocol in each host is said to have local significanc and the routing of each datagram is carried out on a hop-by-hop basis.

12.3 TCP

The transmission control protocol (TCP) provides two communicating pee application protocols – normally one in a client computer and the other in server computer – with a two-way, reliable data interchange service. Althoug the APDUs associated with an application protocol have a defined structure this is transparent to the two communicating peer TCP protocol entitie which treat all the data submitted by each local application entity as a stream of bytes. The stream of bytes flowing in each direction is then transferred (over the underlying network/internet) from one TCP entity to the other in a reliable way; that is, to a high probability, each byte in the stream flowing in

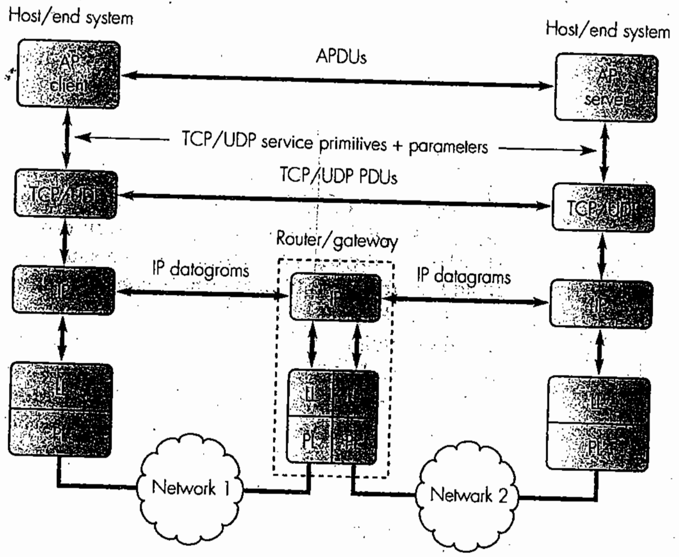


Figure 12.2 TCP/IP protocol suite interlayer communications.

each direction is free of transmission errors, with no lost or duplicate bytes, and the bytes are delivered in the same sequence as they were submitted. The service provided by TCP is known, therefore, as a **reliable stream service**.

As we explained in Section 9.2, the service provided by IP is an unreliable best-effort service. Hence in order to provide a reliable service, as we saw in Section 6.7, before any data is transferred between the two TCP entities, a logical connection is first established between them in order for the sequence numbers in each TCP entity – that are required for error correction and flow control purposes – to be initialized. Also, once all data has been transferred in both directions, the logical connection is closed.

During the actual data transfer phase, in order for the receiving TCP to detect the presence of transmission errors, each TCP entity divides the submitted stream of bytes into blocks known as **segments**. For interactive applications involving a user at a terminal, a segment may contain just a single byte while for large file transfers, a segment may contain many bytes. There is an agreed **maximum segment size** (**MSS**) used with a connection that is established by the two peer TCP entities during the setting up of the connection. This is such that an acceptable proportion of segments are received by the destination free of transmission errors. The default MSS is 536 bytes although larger sizes can be agreed. Normally, the size chosen is such that no fragmentation is necessary during the transfer of a segment over the network/internet and hence is determined by the path MTU. The TCP protocol then includes a retransmission procedure in order to obtain error-free copies of those segments that are received with transmission errors.

In addition, the TCP protocol includes a flow control procedure to ensure no data is lost when the TCP entity in a fast host – a large server for example – is sending data to the TCP in a slower host such as a PC. It also includes a congestion control procedure which endeavors to control the rate of entry of segments into the network/internet to the rate at which segments are leaving.

In the following subsections we discuss firstly the user services provided by TCP, then selected aspects of the operation of the TCP protocol, and finally the formal specification of the protocol. Collectively these are defined in **RFCs 793, 1122**, and **1323**.

12.3.1 User services

The most widely used set of user service primitives associated with TCP are the **socket primitives** used with Berkeley Unix. Hence, although there are a number of alternative primitives, in order to describe the principles involved, we shall restrict our discussion to these. They are operating system calls and collectively form what is called an **application program interface** (**API**) to the underlying TCP/IP protocol stack. A typical list of primitives is given in Table 12.1 and their use is shown in diagrammatic form in Figure 12.3.

Table 12.1 List of socket primitives associated with TCP and thei parameters.

Primitive	Parameters
socket( )	service type, protocol, address format return value = socket descriptor or error code
bind( )	socket descriptor, socket address (= local IP address [illegible] return value = success or error code
listen( )	socket descriptor, maximum queue length return value = success or error code
accept( )	socket descriptor, socket address return value = success or error code
connect( )	socket descriptor, local port number, destination [illegible] destination IP address, precedence, optional data [illegible] user name and a password) return value = success or error code
send( )	socket descriptor, pointer to message buffer [illegible] send, data length (in bytes), push flag, urgent flag return value = success or error code
receive( )	socket descriptor, pointer to a message buffer [illegible] should be put, length of the buffer return value = success or end of file [illegible]
close( )	socket descriptor return value = success or error code
shutdown( )	socket descriptor return value = success or error code

Each of the two peer user application protocols/processes (APs) first cr ates a communications channel between itself and its local TCP entity. This called a **socket** or **endpoint** and, in the case of the server AP, involves the A issuing a sequence of primitive (also known as system of function) calls each with a defined set of parameters associated with it: *socket( ), bind( ), listen( ), accept( )*. Each call has a return value(s) or an error code associated with it.

The parameters associated with the *socket( )* primitive include the service required (reliable stream service), the protocol (TCP), and the address format (Internet). Once a socket has been created – and send/receive memory buffers allocated – a **socket descriptor** is returned to the AP which it then uses with each of the subsequent primitive calls. The AP then issues a

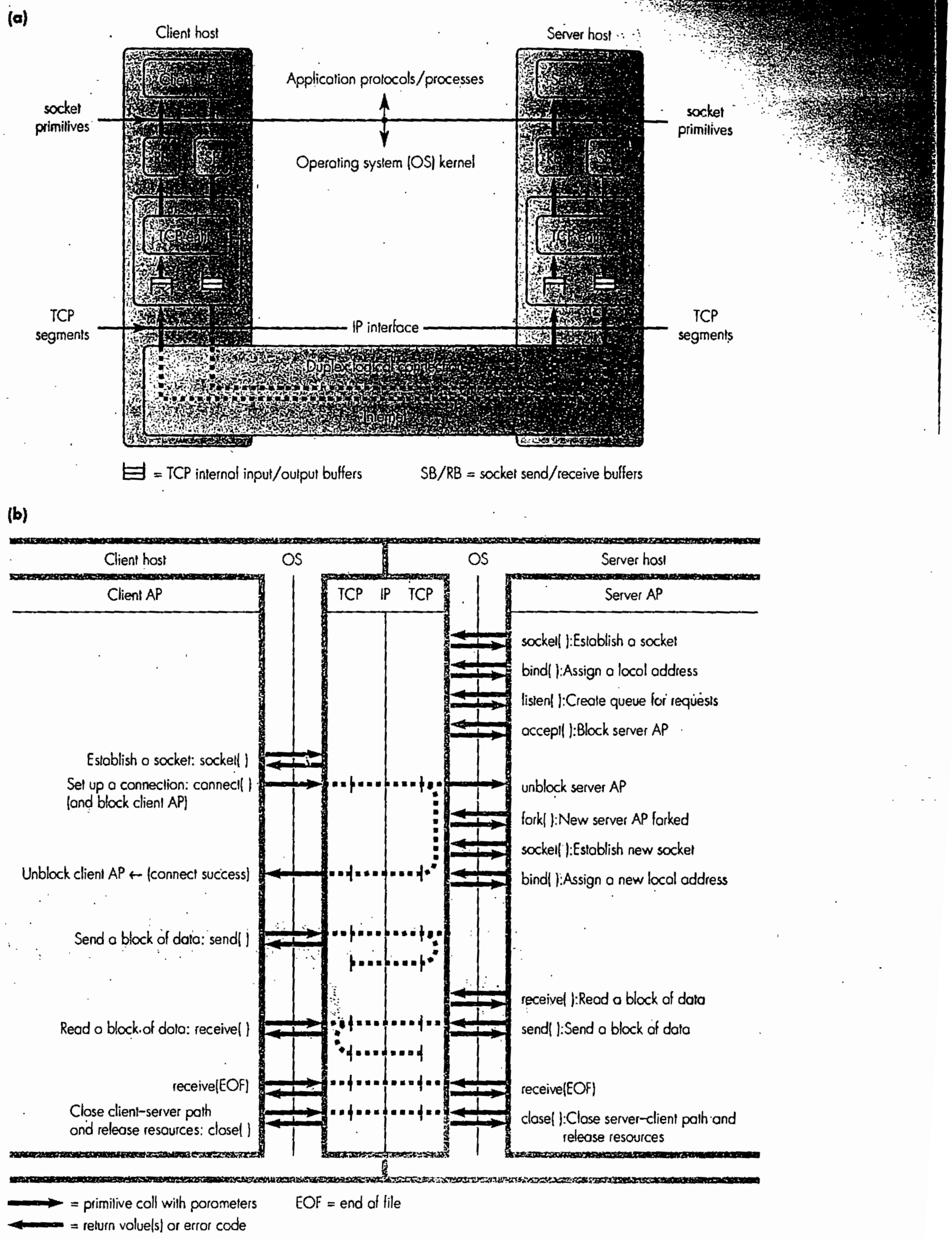

(a)
Client host
Server host
Application protocols/processes
Operating system (OS) kernel
socket primitives
socket primitives
TCP segments
IP interface
TCP segments
Duplex logical connection
= TCP internal input/output buffers
SB/RB = socket send/receive buffers
(b)
Client host
OS
OS
Server host
Client AP
TCP IP TCP
Server AP
socket( ):Establish a socket
bind( ):Assign a local address
listen( ):Create queue for requests
accept( ):Block server AP
Establish a socket: socket( )
Set up a connection: connect( ) (and block client AP)
unblock server AP
fork( ):New server AP forked
socket( ):Establish new socket
bind( ):Assign a new local address
Unblock client AP ← (connect success)
Send a block of data: send( )
receive( ):Read a block of data
Read a block of data: receive( )
send( ):Send a block of data
receive(EOF)
receive(EOF)
Close client-server path and release resources: close( )
close( ):Close server–client path and release resources
= primitive call with parameters
= return value(s) or error code
EOF = end of file

Figure 12.3 TCP socket primitives: (a) socket interface; (b) primitives and their use.

bind( ) primitive which, in addition to the socket descriptor, has an addre parameter associated with it. This is the address the AP wishes to be assigned to the newly created socket and is called the **socket address**. This comprise the Internet-wide IP address of the host and, in the case of the server AP, th 16-bit well-known port number associated with this type of application protocol (FTP and so on).

The *listen( )* primitive call results in the local TCP entity creating a queu (whose maximum length is given as a parameter) to hold incoming connection requests for the server AP. The *accept( )* primitive is then used to put th AP in the blocked state waiting for an incoming connection request to b received from a client TCP entity. Collectively, this sequence of four primitives forms what is called a **passive-open**.

In the case of a client AP, since it can only set up a single TCP connectio at a time and the socket address has only local significance, it simply issues *socket( )* primitive to create a new socket with the same parameters as those used by the server (AP). This is then followed by a *connect( )* primitive which in addition to the locally allocated socket descriptor, has parameters that contain the IP address of the remote (server) host, the well-known port numbe of the required server AP, the local port number that has been assigned t this socket by the client AP, a precedence value, and an optional item of data such as a user name and a password.

The local port number, together with the host IP address, forms the address to be assigned to this socket. The precedence parameter is a collection of parameters that enable the IP protocol to specify the contents of the *type of service* field in the header of the IP datagram that is used to transfer the segments associated with the connection over the Internet. We identified the contents of this field in Figure 9.3 when we discussed the operation of the IP protocol. Note that the IP address of the remote (server) host and the precedence parameters are used by the IP protocol and not TCP. They are examples of what are called **pass-through parameters**; that is, a parameter that is passed down from one protocol layer to another without modification

Once the *connect( )* call has been made, this results in the calling AP being put into the blocked state while the local TCP entity initiates the setting up o a logical connection between itself and the TCP entity in the server Collectively, these two primitives form what is called an **active-open**.

The TCP entity in a client host may support multiple concurrent connections involving different user APs. Similarly, the TCP entity in a server may support multiple connections involving different clients. Hence in order for the two TCP entities to relate each received segment to the correct connection when each new connection is established, both TCP entities create a **connection record** for it. This is a data structure that contains a *connection identifier* (comprising the pair of socket addresses associated with the connection), the agreed *MSS* for the connection, the *initial sequence number* (associated with the acknowledgment procedure) to be used in each direction, the *precedence value*, and the *size of the window* associated with the TCP flow control procedure. Also, a number of fields associated with the operation of the protocol entity including *state variables* and the current state of the protocol entity.

At the server side, when a new connection request (PDU) is received, the server AP is unblocked and proceeds to create a new instance of the server AP to service this connection. Typically, this is carried out using the Unix *fork primitive*. A new socket between the new AP and the local TCP entity is then created and this is used to process the remaining primitives associated with this connection. The parent server AP then either returns to the blocked state waiting for a new connection request to arrive or, if one is already waiting in the server queue, proceeds to process the new request. Once a new instance of the server AP has been created and linked to its local TCP entity by a socket, both the client and server APs can then initiate the transfer of blocks of data in each direction using the *send( )* and *receive( )* primitives.

Associated with each socket is a *send buffer* and a *receive buffer*. The send buffer is used by the AP to transfer a block of data to its local TCP entity for sending over the connection. Similarly, the receive buffer is used by the TCP entity to assemble data received from the connection ready for reading by its local AP. The *send( )* primitive is used by an AP to transfer a block of data of a defined size to the send buffer associated with the socket ready for reading by its local TCP entity. The parameters include: the local socket descriptor, a pointer to the memory buffer containing the block of data, and also the length (in bytes) of the block of data. With TCP there is no correlation between the size of the data block(s) submitted (by an AP to its local TCP entity for sending) and the size of the TCP segments that are used to transfer the data over the logical connection. As we saw in Section 12.2, normally the latter is determined by the path MTU and, in many instances, this is much smaller than the size of the data blocks submitted by an AP.

With some applications, however, each submitted data block may be less than the path MTU. For example, in an interactive application involving a user at a keyboard interacting with a remote AP, the input data may comprise just a few bytes/characters. So in order to avoid the local TCP entity waiting for more data to fill an MTU, the user AP can request that a submitted block of data is sent immediately. This is done by setting a parameter associated with the *send( )* primitive called the *push flag*. A second parameter called the *urgent flag* can also be set by a user AP. This again is used with interactive applications to enable, for example, a user AP to abort a remote computation that it has previously started. The (urgent) data – string of characters – associated with the abort command are submitted by the source AP with the urgent flag set. The local TCP entity then ceases waiting for any further data to be submitted and sends what is outstanding, together with the urgent data, immediately. On receipt of this, the remote TCP entity proceeds to interrupt the peer user AP which then reads the urgent data and acts upon it.

Finally, when a client AP has completed the transfer of all data blocks associated with the connection, it initiates the release of its side of the connection by issuing a *close( )* – or sometimes a *shutdown( )* – primitive. When the server AP is informed of this (by the local TCP entity), assuming it also has finished sending data, it responds by issuing a *close( )* primitive to release the other side of the connection. Both TCP entities then delete their connection records and also the server AP that was forked to service the connectio we shall expand upon later, the *shutdown( )* primitive is used when only ha the connection is to be closed.

12.3.2 Protocol operation

As we can see from the above, the TCP protocol involves three distinct op tions: setting up a logical connection between two previously-created soc transferring blocks of data reliably over this connection, and closing do the logical connection. In practice, each phase involves the exchange of o or more TCP segments (PDUs) and, since all segments have a common st ture, before describing the three phases we first describe the usage of th fields present in each segment header.

Segment format

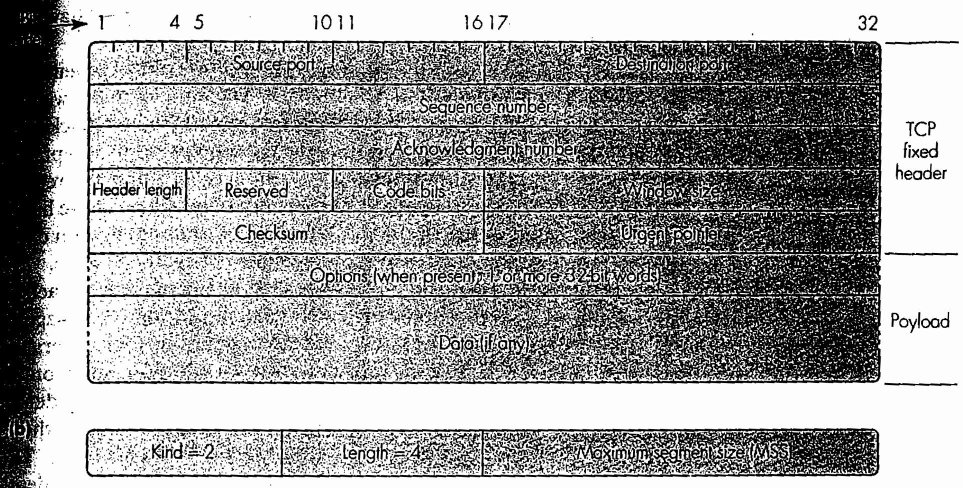
All segments start with a common 20-byte header. In the case of acknowled ment and flow control segments, this is all that is present. In the case connection-related segments, an options field may be present and a data fie is present when data is being transferred over a connection. The field making up the header are shown in Figure 12.4(a).

The 16-bit *source port* and *destination port* fields are used to identify th source and destination APs at each end of a connection. Also, together wi the 32-bit source and destination IP addresses of the related hosts, they fo the 48-bit socket address and the 96-bit connection identifier. Normally, th port number in a client host is assigned by the client AP while the po number in a server is a well-known port.

The *sequence number* performs the same function as the send sequen number in the HDLC protocol and the *acknowledgment number* the same fun tion as the receive sequence number. Also, as with HDLC, a logic connection involves two separate flows, one in each direction. Hence th *sequence number* in a segment relates to the flow of bytes being transmitted b TCP entity and the *acknowledgment number* relates to the flow of bytes in th reverse direction. However, with the TCP, although data is submitted f transfer in blocks, the flow of data in each direction is treated simply a stream of bytes for error and flow control purposes. Hence the *sequence* an *acknowledgment numbers* are both 32-bits in length and relate to the position o a byte in the total session stream rather than to the position of a messag block in the sequence. The *sequence number* indicates the position of the fir byte in the *data* field of the segment relative to the start of the byte stream while the *acknowledgment number* indicates the byte in the stream flowing i the reverse direction that the TCP entity expects to receive next.

The presence of an *options* field in the segment header means that the header can be of variable length. The *header length* field indicates the number of 32-bit words in the header. The 6-bit *reserved* field, as its name implies, is reserved for possible future use.

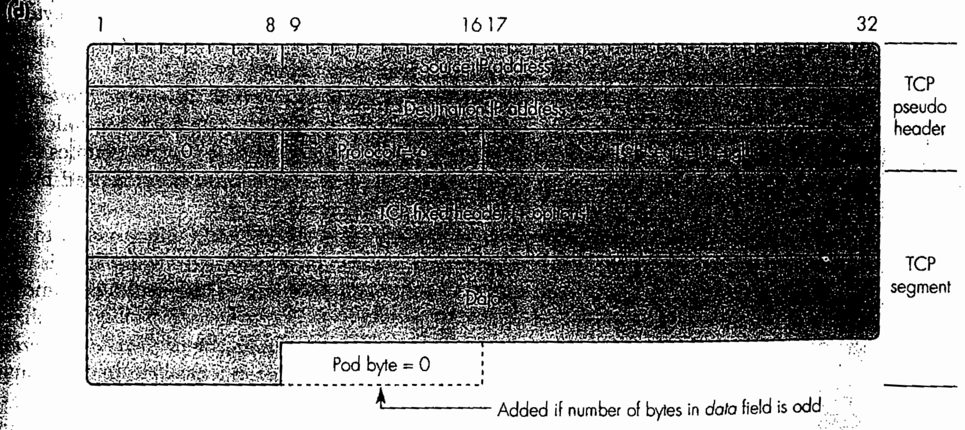
All segments have the same header format and the validity of selected fields in the segment header is indicated by the setting of individual bits in



(a)
1
4 5
10 11
16 17
32
Source port
Destination port
Sequence number
Acknowledgment number
Header length
Reserved
Code bits
Window size
Checksum
Urgent pointer
Options (when present) 1 or more 32-bit words
Data (if any)
TCP fixed header
Payload
(b)
Kind = 2
Length = 4
Maximum segment size (MSS)

(c)

Code bits:	Bit position	Name – function
	11	URG – Urgent pointer field valid
	12	ACK – Acknowledgment field vaild
	13	PSH – Deliver data on receipt of this segment
	14	RST – Reject a connection request or reset sequence numbers to their initial values
	15	SYN – Used to establish a connection
	16	FIN – Used to close a connection



(d)
1
8 9
16 17
32
Source IP address
Destination IP address
0
Protocol = 6
TCP segment length
TCP fixed header + options
Data
Pad byte = 0
Added if number of bytes in data field is odd
TCP pseudo header
TCP segment

Figure 12.4 TCP segment format: (a) header fields; (b) MSS option format; (c) code bit definitions; (d) pseudo header fields.

the 6-bit *code* field; if a bit is set (a binary 1), the corresponding field is va Note that multiple bits can be set in a single segment. The bits have meaning shown in Figure 12.4(c).

The *window size* field relates to a sliding window flow control scheme principles of which we considered in Section 6.7.4. The number in the wind size indicates the number of data bytes (relative to the byte being ackno edged in the *acknowledgment* field) that the receiver is prepared to accept determined by the amount of unused space in the receive buffer the remo TCP entity is using for this connection. The maximum size of the receive buff – and hence the maximum size of the window – can be different in each dire tion and has a default value which, typically, is 4096, 8192, or 16384 bytes.

As we saw in Section 9.2, the checksum field in the header of each datagram applies only to the fields in the IP header and not the datagram contents. Hence the *checksum* field in the TCP segment header covers th complete segment; that is, header plus contents. In addition, since only simple checksum is used to derive the checksum value in the IP header, in order to add an additional level of checking, some selected fields from the I header are also included in the computation of the TCP checksum. The fields used form what is called the (TCP) **pseudo header** and these are identi fied in Figure 12.4(d).

As we can see, these are the source and destination IP addresses and th protocol value (= 6 for TCP) from the IP header, plus the total byte count o the TCP segment (header plus contents). The computation of the checksum uses the same algorithm as that used by IP. As we saw in Section 9.2, this i computed by treating the complete datagram as being made up of a string o 16-bit words which are then added together using 1s complement arithmeti Since the number of bytes in the original TCP segment *data* field may be odd in order to ensure the same checksum is computed by both TCP entities, a **pad byte** of zero is added to the data field whenever the number of bytes in the original *data* field is odd. As we can deduce from this, the byte count o the TCP segment must always be an even integer.

When the URG (urgent) flag is set in the *code* field, the *urgent pointer* fiel is valid. This indicates the number of bytes in the *data* field that follow the current *sequence number*. This is known as **urgent data** – or sometimes **expe dited data** – and, as we mentioned earlier, it should be delivered by the receiving TCP entity immediately it is received.

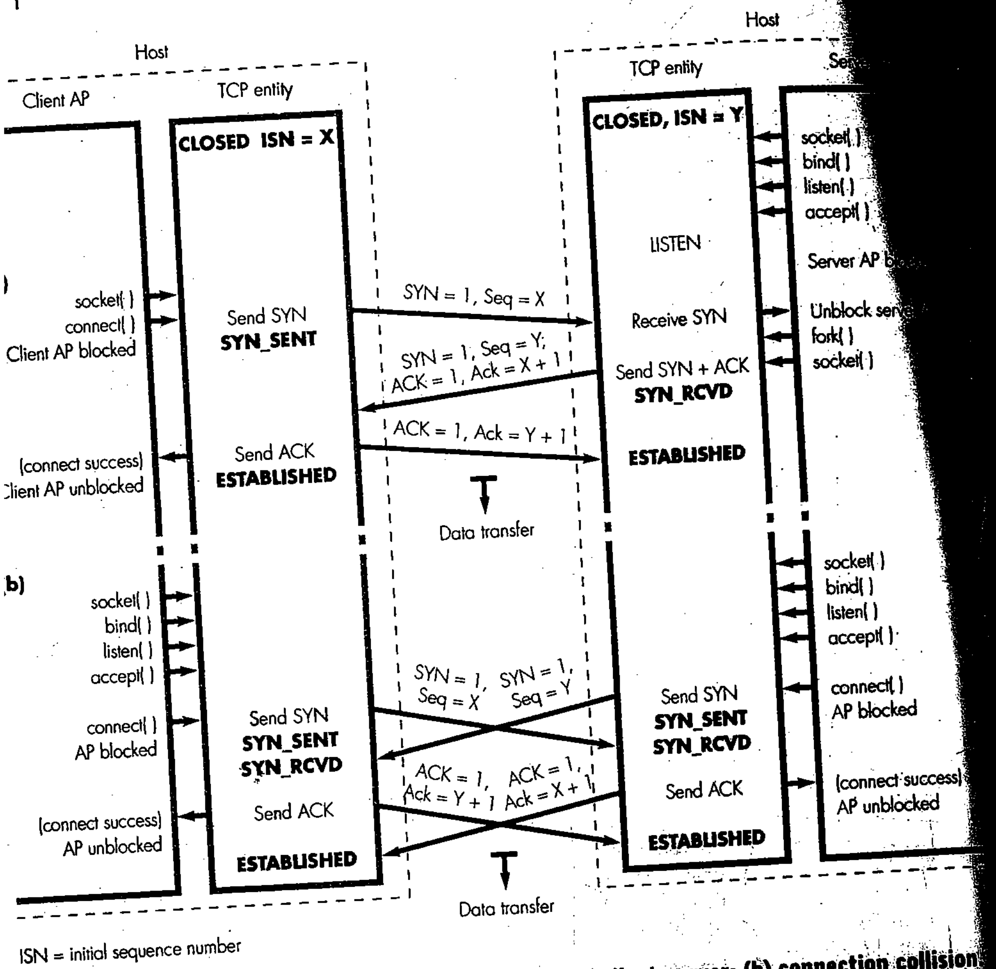
The *options* field provides the means of adding extra functionality to that covered by the various fields in the segment header. For example, it is used during the connection establishment phase to agree the maximum amount of data in a segment each TCP entity is prepared to accept. During this phase, each indicates its own preferred maximum size and hence can be different for each direction of flow. As we indicated earlier, this is called the maximum segment size (MSS) and excludes the fixed 20-byte segment header. If one of the TCP entities does not specify a preferred maximum size then a default value of 536 bytes is chosen. The TCP entity in all hosts connected to the Internet must accept a segment of up to 556 bytes – 536 plus a 20-byte header – and all IPs must accept a datagram of 576 bytes – 556 bytes plus a further 20-byte (IP) header. Hence the default MSS of 536 bytes ensures the datagram (with the TCP segment in its payload) will be transferred over the Internet without fragmentation. The format of the MSS option is shown in Figure 12.4(b). Also, although not shown, if this is the last or only option present in the header, then a single byte of zero is added to indicate this is the end of the option list.

Connection establishment

On receipt of a *connect( )* primitive (system call) from a client AP, the local TCP entity attempts to set up a logical connection with the TCP entity in the server whose IP address (and port number) are specified in the parameters associated with the primitive. This is achieved using a three-way exchange of segments. Collectively, this is known as a **three-way handshake** procedure and the segments exchanged for a successful *connect( )* call are shown in Figure 12.5(a).

As we indicated in the previous section, the flow of data bytes in each direction of a connection is controlled independently. The TCP entity at each end of a connection starts in the CLOSED state and chooses its own *initial sequence number* (*ISN*). These are both non-zero and change from one connection to another. This ensures that any segments relating to a connection that get delayed during their transfer over the internet – and hence arrive at the client/server after the connection has been closed – do not interfere with the segments relating to the next connection. Normally, each TCP entity maintains a separate 32-bit counter that is incremented at intervals of either 4 or 8 microseconds. Then, when a new ISN is required, the current contents of the counter are used.

- To establish a connection, the TCP at the client first reads the ISN to be used (from the counter) and makes an entry of this in the *ISN* and *send sequence variable* fields of the connection record used for this connection. It then sends a segment to the TCP in the server with the SYN code bit on, the ACK bit off, and the chosen ISN (X) in the sequence (number) field. Note that since no window or MSS option fields are present, then the receiving TCP assumes the default values. The TCP entity then transfers to the SYN_SENT state.
- On receipt of the SYN, if the required server AP – as determined by the destination port and IP address – is not already in the LISTEN state, the server TCP declines the connection request by returning a segment with the RST code bit on. The client TCP entity then aborts the connection establishment procedure and returns an error message with a reason code to the client AP. Alternatively, if the server AP is in the LISTEN state, the server TCP makes an entry of the ISN (to be used in the client–server direction and contained within the received segment) in its own connection record – in both the *ISN* and *receive sequence variable* fields – together with the ISN it proposes to use in the return direction. It then proceeds to create a new segment with the SYN bit on and the chosen ISN(Y) in the sequence field. In addition, it sets the ACK bit on


Host
Client AP
TCP entity
CLOSED ISN = X
socket( )
connect( )
Client AP blocked
Send SYN
SYN_SENT
(connect success)
Client AP unblocked
Send ACK
ESTABLISHED
SYN = 1, Seq = X
SYN = 1, Seq = Y;
ACK = 1, Ack = X + 1
ACK = 1, Ack = Y + 1
Data transfer
Host
TCP entity
CLOSED, ISN = Y
socket( )
bind( )
listen( )
accept( )
LISTEN
Server AP b
Receive SYN
Unblock serv
fork( )
socket( )
Send SYN + ACK
SYN_RCVD
ESTABLISHED
b)
socket( )
bind( )
listen( )
accept( )
connect( )
AP blocked
Send SYN
SYN_SENT
SYN_RCVD
(connect success)
AP unblocked
Send ACK
ESTABLISHED
SYN = 1, Seq = X
SYN = 1, Seq = Y
ACK = 1, Ack = Y + 1
ACK = 1, Ack = X + 1
Data transfer
socket( )
bind( )
listen( )
accept( )
connect( )
AP blocked
Send SYN
SYN_SENT
SYN_RCVD
Send ACK
ESTABLISHED
(connect success)
AP unblocked

ISN = initial sequence number

Figure 12.5 TCP connection establishment examples: (a) client–server; (b) connection collision

and returns a value of X+1 in the acknowledgment field to acknowledge receipt of the client's SYN. It then sends the segment to the client and enters the SYN_RCVD state.

- On receipt of the segment, the client TCP makes an entry of the ISN to be used in the server–client direction in its connection record – in both the *ISN* and *receive sequence variable* fields – and increments the send sequence variable in the record by 1 to indicate the SYN has been acknowledged. It then acknowledges receipt of the SYN by returning a

segment with the ACK bit on and a value of Y+1 in the acknowledgment field. The TCP entity then enters the (connection) ESTABLISHED state.

- On receipt of the ACK, the server TCP increments the send sequence variable in its own connection record by 1 and enters the ESTABLISHED state. As we can deduce from this, the acknowledgment of each SYN segment is equivalent to a single data byte being transferred in each direction. At this point, both sides are in the ESTABLISHED state and ready to exchange data segments.

Although in a client–server application the client always initiates the setting up of a connection, in applications not based on the client–server model the two APs may try to establish a connection at the same time. This is called a **simultaneous open** and the sequence of segments exchanged in this case are as shown in Figure 12.5(b).

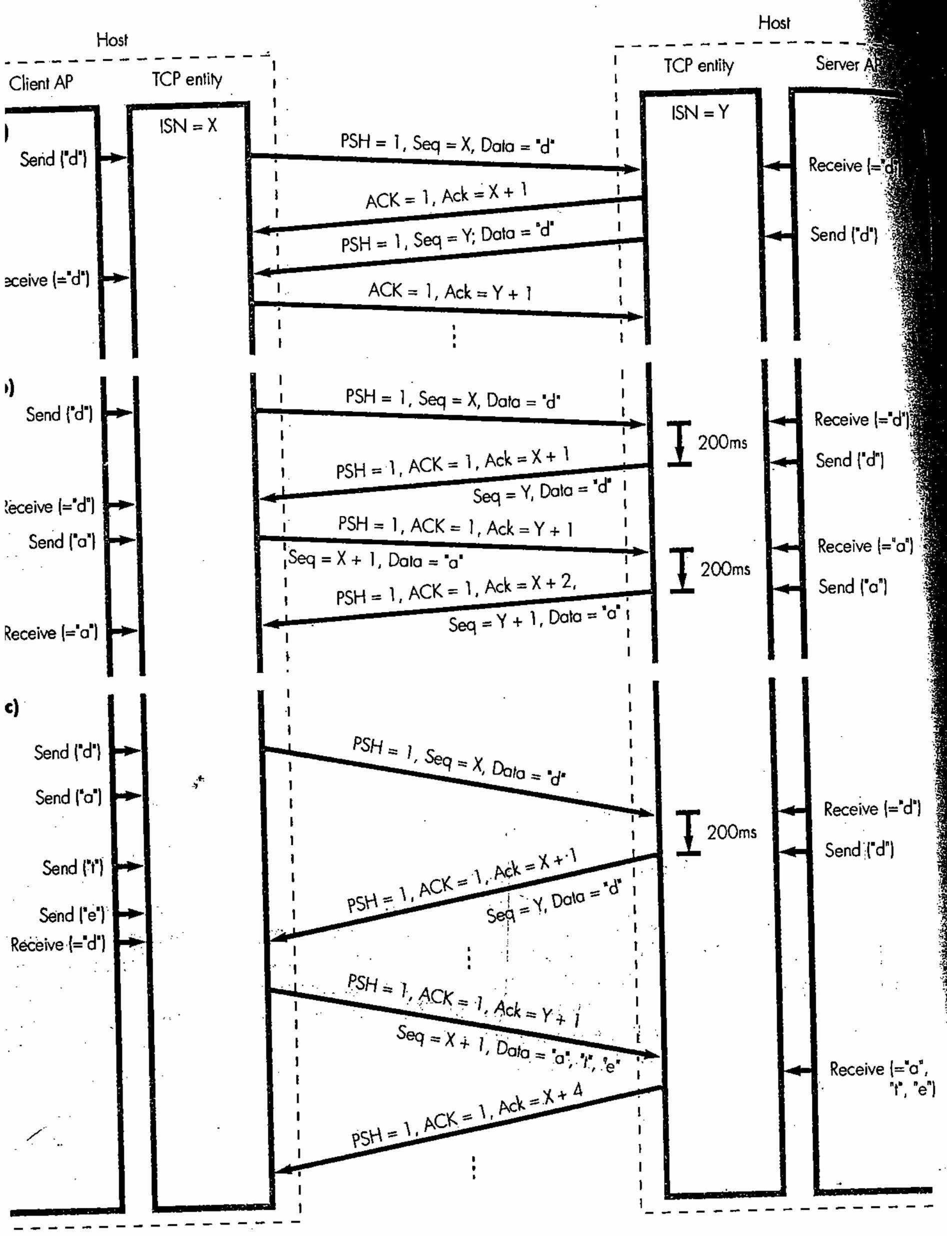
As we can see, the segments exchanged are similar to those in the client–server case and, since both ISNs are different, each side simply returns a segment acknowledging the appropriate sequence number. However, since the connection identifier is the same in both cases, only a single connection is established.

Data transfer

The error control procedure associated with the TCP protocol is similar to that used with the HDLC protocol we described earlier in Section 6.8. The main difference is that the sequence and acknowledgment numbers used with TCP relate to individual bytes in the total byte stream whereas with HDLC the corresponding send and receive sequence numbers relate to individual blocks of data. Also, because of the much larger round-trip time of an internet (compared with a single link), with TCP the window size associated with the flow control procedure is not derived from the sequence numbers. Instead, a new window size value is included in each segment a TCP entity sends to inform the other TCP entity of the maximum number of bytes it is willing to receive from it. This is known also as a **window size advertisement.**

In addition, because the TCP protocol may be operating (on an end-to-end basis) over a number of interconnected networks rather than a single line, it includes a congestion control procedure. This endeavors to regulate the rate at which the sending TCP entity sends data segments into the internet to the rate segments are leaving the internet. We shall discuss the main features of the different procedures that are used by means of examples.

Small segments In order to explain the features of the protocol that relate to the exchange of small segments – that is, all the segments contain less than the MSS – we shall consider a typical data exchange relating to a networked interactive application. An example application protocol of this type is Telnet. Typically, this involves a user at the client side typing a command and the server AP in a remote host responding to it. An example set of segments is shown in Figure 12.6(a).


Host
Client AP
TCP entity
ISN = X
Host
TCP entity
Server A
ISN = Y
Send ("d")
PSH = 1, Seq = X, Data = "d"
Receive (="d
ACK = 1, Ack = X + 1
Send ("d")
PSH = 1, Seq = Y; Data = "d"
eceive (="d")
ACK = 1, Ack = Y + 1
Send ("d")
PSH = 1, Seq = X, Data = "d"
Receive (="d")
200ms
Send ("d")
PSH = 1, ACK = 1, Ack = X + 1
Seq = Y, Data = "d"
eceive (="d")
Send ("a")
PSH = 1, ACK = 1, Ack = Y + 1
Seq = X + 1, Data = "a"
Receive (="a")
200ms
Send ("a")
PSH = 1, ACK = 1, Ack = X + 2,
Seq = Y + 1, Data = "a"
Receive (="a")
c)
Send ("d")
Send ("a")
PSH = 1, Seq = X, Data = "d"
Receive (="d")
200ms
Send ("d")
Send ("t")
PSH = 1, ACK = 1, Ack = X + 1
Seq = Y, Data = "d"
Send ("e")
Receive (="d")
PSH = 1, ACK = 1, Ack = Y + 1
Seq = X + 1, Data = "a", "t", "e"
Receive (="a",
"t", "e")
PSH = 1, ACK = 1, Ack = X + 4

gure 12.6 Small segment data transfers: (a) immediate acknowledgments; (b) delayed knowledgments; (c) Nagle algorithm.

With interactive applications involving a user at a keyboard, each character typed is sent directly by the client AP to the server side. The server AP then reads the character from the receive buffer and immediately echoes the character back to the client side by writing it into the send buffer. On receipt of the character, the client AP displays it on the host screen. Hence each typed character is sent directly in a separate segment with the PSH flag on. Similarly, the echoed character is also sent in a separate segment with the PSH flag on. In addition to these two segments, however, each TCP entity returns a segment with the ACK bit on to acknowledge receipt of the segment containing the typed/echoed character. This means that for each character that is typed, four segments are sent, each with a 20-byte header and a further 20-byte IP header.

In practice, in order to reduce the number of segments that are sent, a receiving TCP entity does not return an ACK segment immediately it receives an (error-free) data segment. Instead, it waits for up to 200 ms to see if any data is placed in the send buffer by the local AP. If it is, then the acknowledgment is piggybacked in the segment that is used to send the data. This procedure is called **delayed acknowledgements** and, as we can see in Figure 12.6(b), with interactive applications it can reduce significantly the number of segments that are required.

Although this mode of working is acceptable when communicating over a single network such as a LAN, when communicating over an internet which has a long round-trip time (RTT), the delays involved in waiting for each echoed character can be annoying to the user. Hence a variation of the basic delayed acknowledgment procedure is often used. This is called the **Nagle algorithm** and is defined in **RFC 896**. When the algorithm is enabled, each TCP entity can have only a single small segment waiting to be acknowledged at a time. As a result, in interactive applications, when the client TCP entity is waiting for the ACK for this segment to be received, a number of characters may have been typed by the user. Hence when the ACK arrives, all the waiting characters in the send buffer are transmitted in a single segment. A sequence diagram showing this is given in Figure 12.6(c).

In these examples, the window size has not been shown since, in general, when small segments are being exchanged it has no effect on the flow of the segments. Also, the Nagle algorithm is not always enabled. For example, when the interactions involve a mouse, each segment may contain a collection of mouse movement data and, when echoed, the movement of the cursor can appear erratic. An example application of this type is X-Windows.

Error control As we saw in Figure 12.4(a), each TCP segment contains only a single acknowledgment number. Hence should a segment not arrive at the destination host, the receiving TCP can only return an acknowledgment indicating the next in-sequence byte that it expects. Also, since the packets relating to a message are being transmitted over an internet, when the path followed has alternative routes, packets may arrive at the destination host out of sequence. Hence the receiving TCP simply holds each out-of-sequenc ment that it receives in temporary storage and returns an ACK indicatin next in-sequence byte – and hence segment – that it expects. Normall out-of-sequence segment arrives within a short time interval at which po the receiving TCP returns an ACK that acknowledges all the segments received including those held in temporary storage.

At the sending side, the TCP receives an ACK indicating a segment been lost but, within a short time interval, it receives an ACK for a segm that it transmitted later (so acknowledging receipt of all bytes up to a including the last byte in the later segment). Hence, since this is a relativ frequent occurrence, the sending TCP does not initiate a retransmissio immediately it receives an out-of-sequence ACK. Instead, it only retransmit segment if it receives three duplicate ACKs for the same segment – that three consecutive segments with the same acknowledgment number in the header – since it is then confident that the segment has been lost rather th simply received out of sequence.

In addition, to allow for the possibility that the sending TCP has no fu ther segments ready for transmission, when a loss is detected it starts *retransmission timer* for each new segment it transmits. A segment is the retransmitted if the TCP does not receive an acknowledgment for it befor the timeout interval expires. An example illustrating the two possibilities shown in Figure 12.7. In the example we assume:

- there is only a unidirectional flow of data segments;
- the sending AP writes a block of data – a message – comprising 3072 bytes into the send buffer using a *send( )* primitive;
- the MSS being used for the connection is 512 bytes and hence 6 segments are required to send the block of data;
- the size of the receive buffer is 8192 bytes and hence the sending TCP can send the complete set of segments without waiting for an acknowledgment;
- an ACK segment is returned on receipt of each error-free data segment;
- segments 2 and 6 are both lost – owing to transmission errors for example – as is the final ACK segment.

To follow the transmission sequence shown in the figure we should note the following:

- The sending TCP has a send sequence variable, V(S), in its connection record which is the value it places in the sequence number field of the next new segment it sends. Also a *retransmission list* to hold segments waiting to be acknowledged. Similarly, the receiving TCP has a receive sequence variable, V(R), in its connection record (which is the sequence number it expects to receive next) and a *receive list* to hold segments that are received out of sequence.

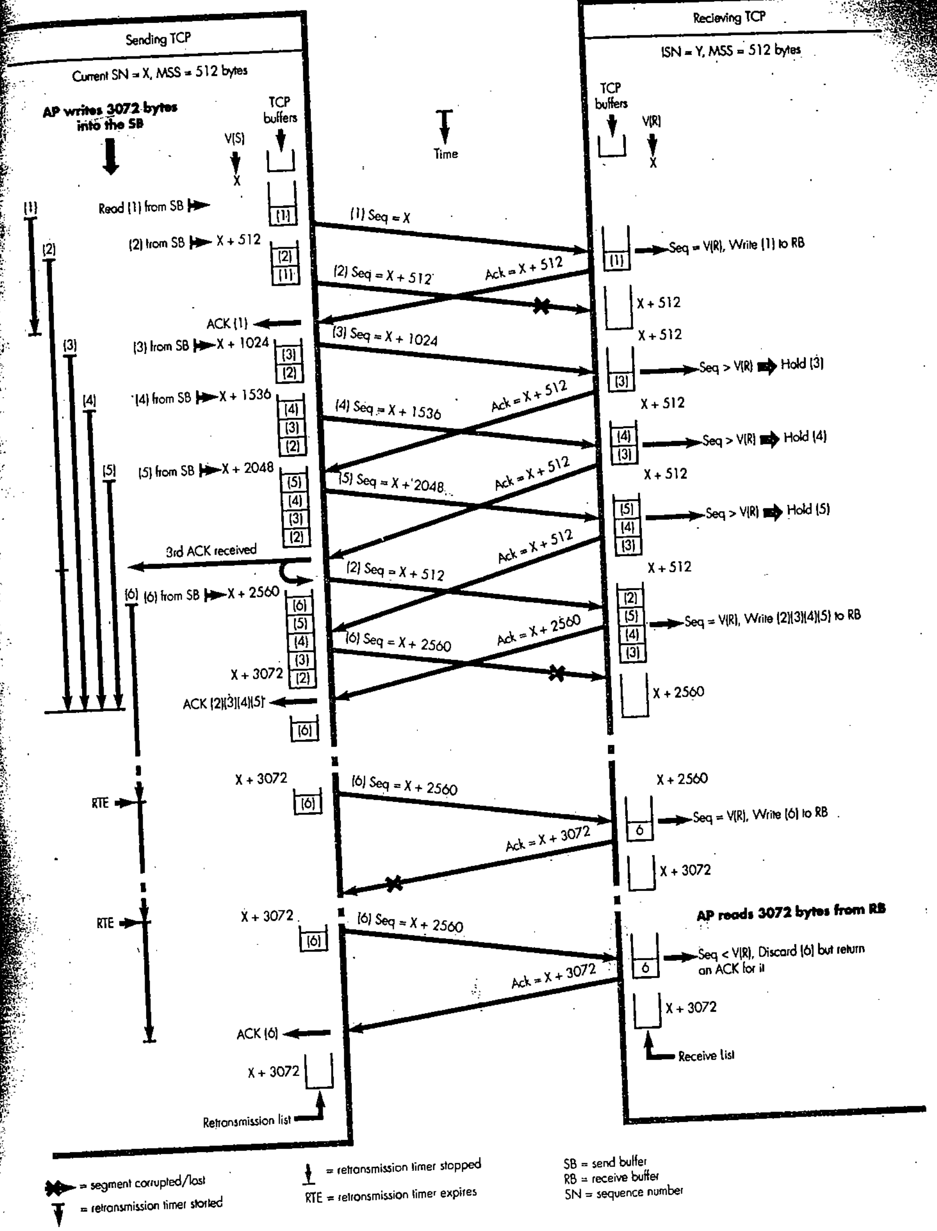

Sending TCP
Current SN = X, MSS = 512 bytes
AP writes 3072 bytes into the SB
TCP buffers
V(S)
X
Time
Recieving TCP
ISN = Y, MSS = 512 bytes
TCP buffers
V(R)
X
(1)
(2)
(3)
(4)
(5)
(6)
Read (1) from SB
(2) from SB X + 512
ACK (1)
(3) from SB X + 1024
(4) from SB X + 1536
(5) from SB X + 2048
3rd ACK received
(6) from SB X + 2560
X + 3072
ACK (2)(3)(4)(5)
(1) Seq = X
(2) Seq = X + 512
(3) Seq = X + 1024
(4) Seq = X + 1536
(5) Seq = X + 2048
(2) Seq = X + 512
(6) Seq = X + 2560
Ack = X + 512
Ack = X + 2560
Ack = X + 3072
Seq = V(R), Write (1) to RB
X + 512
Seq > V(R) Hold (3)
Seq > V(R) Hold (4)
Seq > V(R) Hold (5)
Seq = V(R), Write (2)(3)(4)(5) to RB
X + 2560
Seq = V(R), Write (6) to RB
X + 3072
AP reads 3072 bytes from RB
Seq < V(R), Discard (6) but return an ACK for it
Receive list
RTE
ACK (6)
Retransmission list
= segment corrupted/lost
= retransmission timer started
= retransmission timer stopped
RTE = retransmission timer expires
SB = send buffer
RB = receive buffer
SN = sequence number

Figure 12.7 Example segment sequence showing TCP error control procedure.

- Segment (1) is received error-free and, since its sequence number (Seq = X) is equal to V(R), the 512 bytes of data it contains are transferred directly into the receive buffer (ready for reading by the destination AP), V(R) is incremented to X + 512, and an ACK (with Ack = X + 512) is returned to the sending side.
- On receipt of the ACK, the sending TCP stops the retransmission timer for (1) and, in the meantime, segment (2) has been sent.
- Since segment (2) is corrupted, no ACK for it is returned and hence its timer continues running.
- The sending TCP continues to send segments (3), (4), and (5) all of which are received error-free. However, since they are out of sequence – segment (2) is missing – the receiving TCP retains them in its receive list and returns an ACK with Ack = X + 512 in each segment for each of them to indicate to the sending TCP that segment (2) is missing.
- On receipt of the third ACK with an ACK = X + 512, the sending TCP retransmits segment (2) without waiting for the retransmission timer to expire. As we indicated earlier, this is done since if three or more ACKs with the same acknowledgment number are received one after the other, it is assumed that the segment indicated has been lost rather than received out of sequence. Because the retransmission occurs before the timer expires, this procedure is called **fast retransmit**.
- This time segment (2) is received error-free and, as a result, the receiving TCP is able to transfer the contents of segments (2), (3), (4), and (5) to the receive buffer – ready for reading by the AP – and returns an ACK with Ack = X + 2560 to indicate to the sending TCP that all bytes up to and including byte 2560 have been received and their timers can be stopped.
- In the meantime, segment (6) has been transmitted but is corrupted. Hence, since no other segments are waiting to be sent, its timer continues running until it expires when the segment is retransmitted.
- This time the segment is received error-free and so its contents are passed to the receive buffer directly and an ACK for it is returned. Also, it is assumed that at this point the receiving AP reads the accumulated 3072 bytes from the receive buffer using a *receive( )* primitive.
- The ACK for segment (6) is corrupted and hence the timer for the segment expires again and the segment is retransmitted. However, since the sequence number is less than the current V(R), the receiving TCP assumes it is a duplicate. Hence it discards it but returns an ACK to stop the sending TCP from sending another copy.

As we can see from this example, a key parameter in the efficiency of the error control procedure is the choice of the **retransmission timeout (RTO) interval**. With a single data link the choice of an RTO is straightforward since the worst-case round-trip time – the time interval between sending a packet/frame and receiving an ACK for it – can be readily determined. The RTO is then set at a value slightly greater than this. With an internet, however, the RTT of a TCP connection can vary considerably over relatively short intervals as the traffic levels within routers build up and subside. Hence choosing an RTO when the internet is lightly loaded can lead to a significant number of unnecessary retransmissions, while choosing it during heavy load conditions can lead to unnecessary long delays each time a segment is corrupted/lost. The choice of RTO, therefore, must be dynamic and vary not only from one connection to another but also during a connection.

The initial approach used to derive the RTO for a connection was based on an **exponential backoff** algorithm. With this an initial RTO of 1.5 seconds is used. Should this prove to be too short – that is, each segment requires multiple retransmission attempts – the RTO is doubled to 3 seconds. This doubling process then continues until no retransmissions occur. To allow for network problems, a maximum RTO of 2 minutes is used at which point a segment with the RST flag bit on is sent to abort the connection/session.

Although very simple to implement, a problem with this method is that when an ACK is received, it is not clear whether this is for the last retransmission attempt or an earlier attempt. This is known as the **retransmission ambiguity problem** and was identified by Karn. Because of this, a second approach was proposed by Jacobson and is defined in **RFC 793**. With this method, the RTO is computed from actual RTT measurements. The RTO is then updated as each new RTT measurement is made. In this way, the RTO for each connection is continuously being updated as each new estimate of the RTT is determined.

As we indicated earlier, when each data segment is sent, a separate retransmission timer is started. A connection starts with a relatively large RTO. Then, each time an ACK segment is received before the timer expires, the actual RTT is determined by subtracting the initial start time of the timer from the time when the ACK was received. The current estimate of the RTT is then updated using the formula

$$\mathrm{RTT} = \alpha\,\mathrm{RTT} + (1-\alpha)\,M$$

where M is the measured RTT and α is a smoothing factor which determines by how much each new M influences the computation of the new RTT relative to the current value. The recommended value for α is 0.9. The new RTO is then set at twice the updated RTT to allow for a degree of variance in the RTT.

Although this method performed better than the original method, a refinement of it was later introduced. This was done because by using a fixed multiple of each updated RTT (× 2) to compute the new RTO, it was found that it did not handle well the wide variations that occurred in the RTT. Hence in order to obtain a better estimate, Jacobson proposed that each new RTO should be based not just on the mean of the measured RTT but also on the variance. In the proposed algorithm, the mean deviation of the RTT measurements, D, is used as an estimate of the variance. It is computed using the formula:

$$D = \alpha\,D + (1-\alpha)\,|\mathrm{RTT} - \mathrm{M}|$$

where α is a smoothing factor and |RTT – M| is the modulus of the diffe between the current estimate of the RTT and the new measured RTT This is also computed for each new RTT measurement and the new esti of the RTO is then derived using the formula:

$$RTO = RTT + 4D$$

As we indicated earlier, to overcome the retransmission ambiguity probl the RTT is not measured/updated for the ACKs relating to retransmi segments.

Note also that although in the above example an ACK is returned o receipt of each data segment, this is not always the case. Indeed, in mo implementations, providing there is a steady flow of data segments, th receiving TCP only returns an ACK for every other segment it receives co rectly. On sending each ACK, a timer – called the **delayed ACK timer** – started and, should a second segment not be received before it expires, the an ACK for the first segment is sent. Note, however, that when a single ACK sent for every other segment, since the V(R) is incremented on receipt o each segment, then the acknowledgment number within the (ACK) segmen acknowledges the receipt of all the bytes in both segments.

Flow control As we indicated earlier, the value in the *window size* field of each segment relates to a sliding window flow control scheme and indicates th number of bytes (relative to the byte being acknowledged in the *acknowledgment* field) that the receiving TCP is able to accept. This is determined by th amount of free space that is present in the receive buffer being used by th receiving TCP for the connection. Recall also that the maximum size of th window is determined by the size of the receive buffer. Hence when the sending TCP is running in a fast host – a large server for example – and th receiving TCP in a slow host, the sending TCP can send segments faster than firstly, the receiving TCP can process them and, secondly, the receiving A can read them from the receive buffer after they have been processed. Th window flow control scheme, therefore, is present to ensure that there i always the required amount of free space in the receive buffer at the destination before the source sends the data. An example showing the sequence of segments that are exchanged to implement the scheme is given in Figure 12.8. In the example, we assume:

- there is only a unidirectional flow of data segments;
- the size of both the send and the receive buffers at the sending side are 4096 bytes and those at the receiving side 2048 bytes. Hence the maximum size of the window for the direction of flow shown is 2048 bytes;
- associated with each direction of flow the sending side maintains a *send window variable*, W_S, and the receiving side a *receive window variable*, W_R;

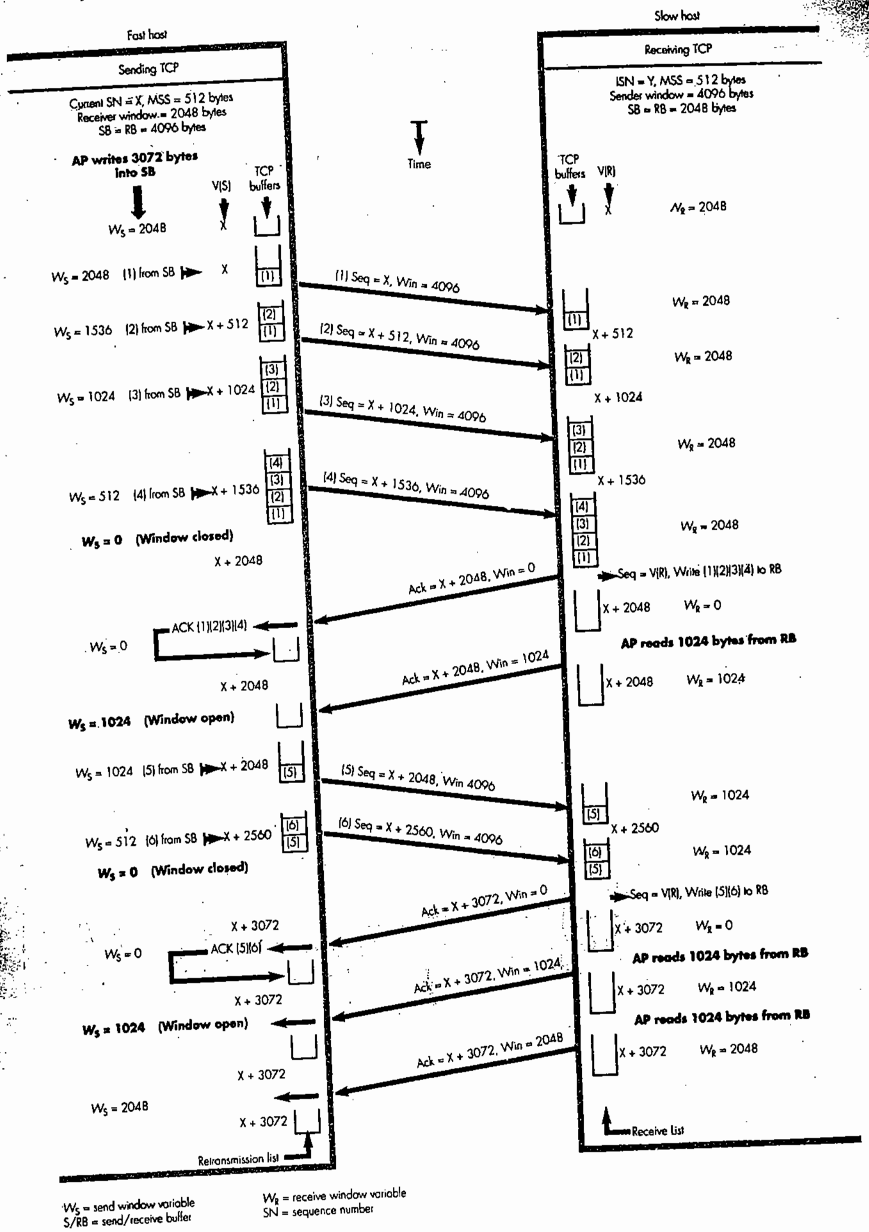

Fast host
Sending TCP
Current SN = X, MSS = 512 bytes
Receiver window = 2048 bytes
SB = RB = 4096 bytes
AP writes 3072 bytes into SB
V(S)
TCP buffers
Time
Slow host
Receiving TCP
ISN = Y, MSS = 512 bytes
Sender window = 4096 bytes
SB = RB = 2048 bytes
TCP buffers
V(R)
W_S = 2048
W_R = 2048
W_S = 2048 (1) from SB X
(1) Seq = X, Win = 4096
W_S = 1536 (2) from SB X + 512
(2) Seq = X + 512, Win = 4096
W_S = 1024 (3) from SB X + 1024
(3) Seq = X + 1024, Win = 4096
W_S = 512 (4) from SB X + 1536
(4) Seq = X + 1536, Win = 4096
W_S = 0 (Window closed)
X + 2048
Seq = V(R), Write (1)(2)(3)(4) to RB
Ack = X + 2048, Win = 0
X + 2048 W_R = 0
ACK (1)(2)(3)(4)
W_S = 0
AP reads 1024 bytes from RB
Ack = X + 2048, Win = 1024
X + 2048 W_R = 1024
W_S = 1024 (Window open)
W_S = 1024 (5) from SB X + 2048
(5) Seq = X + 2048, Win 4096
W_R = 1024
X + 2560
W_S = 512 (6) from SB X + 2560
(6) Seq = X + 2560, Win = 4096
W_S = 0 (Window closed)
Seq = V(R), Write (5)(6) to RB
Ack = X + 3072, Win = 0
X + 3072 W_R = 0
ACK (5)(6)
W_S = 0
AP reads 1024 bytes from RB
Ack = X + 3072, Win = 1024
X + 3072 W_R = 1024
W_S = 1024 (Window open)
AP reads 1024 bytes from RB
Ack = X + 3072, Win = 2048
X + 3072 W_R = 2048
W_S = 2048
Retransmission list
Receive list
W_S = send window variable
S/RB = send/receive buffer
W_R = receive window variable
SN = sequence number

Figure 12.8 Example segment sequence showing TCP flow control procedure.

- the source AP can write bytes into the send buffer up to the current valu of W_S and, providing W_S is greater than zero, the sending TCP can read data bytes from its send buffer up to the current value of W_S and initiate their transmission. Flow is stopped when $W_S = 0$ and the window is then said to be closed;
- at the destination, the receiving TCP, on receipt of error-free data segments, transfers the data they contain to the receive buffer and increments W_R by the number of bytes transferred. W_R is then decremented when the destination AP reads bytes from the receive buffer and, after each read operation, the receiving TCP returns a segment with the number of bytes of free space now available in the window size field of the segment.

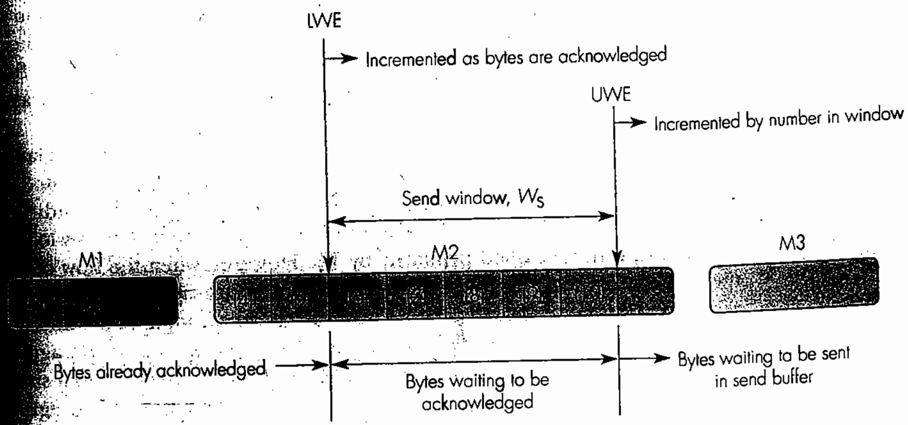
The following should be noted when interpreting the sequence:

- The flow of segments starts with the AP at the sending side writing 3072 bytes into the send buffer using a *send( )* primitive.
- Since the sending host is much faster than the receiving host, the sending TCP is able to send a full window of 2048 bytes (in four 512-byt segments) before the receiving TCP is able to start processing them. The sending TCP must then stop as W_S is now zero.
- When the receiving TCP is scheduled to run, it finds four segments in it receive list and, since the first segment – segment (1) – has a sequence number equal to V(R), it transfers its contents to the receive buffer. It then proceeds to process and transfer the contents of segments (2), (3) and (4) in the same way and, after it has done this, it returns a single ACK segment to acknowledge receipt of these four segments but with a window size field of zero.
- On receipt of the ACK, the sending TCP deletes segments (1), (2), (3) and (4) from its retransmission list but leaves $W_S = 0$.
- When the receiving AP is next scheduled to run, we assume it reads jus 1024 bytes from the receive buffer. On detecting this, the TCP returns second ACK with the same acknowledgment number but with a windo size of 1024. The second ACK is known, therefore, as a **window updat**
- At this point, since its sending window is now open, the sending TCP proceeds to send segments (5) and (6) at which point W_S again becom zero.
- At the receiving side, when the TCP is next scheduled to run it finds segments (5) and (6) in the receive list and, since their sequence numbers are in sequence, it transfers their contents to the receive buffe It then returns a single ACK for them but with a window size of zero.
- On receipt of the ACK, the sending TCP deletes segments (5) and (6) from its retransmission list but leaves $W_S = 0$.

- At some point later, the receiving AP is scheduled to run and we assume it again reads 1024 bytes from the receive buffer. Hence when the TCP next runs it returns a window update of 1024.
- Finally, after the receiving AP reads the last 1024 bytes from the RB, the TCP – some time later – returns a window update of 2048. After this has been received, both sides are back to their initial state.

We should note that there are a number of different implementations of TCP and hence the sequence shown in Figure 12.8 is only an example. For example, the receiving TCP may return two ACKs – one for segments (1) and (2) and the other for segments (3) and (4) – rather than a single ACK. In this case there would be a different distribution of segments between the TCP buffers and the receive buffer. Nevertheless, providing the size of the TCP buffers in the receiver is the same as the receive buffer, then the window procedure ensures there is sufficient buffer storage to hold all received segments. A schematic diagram summarizing the operation of the sliding window procedure is given in Figure 12.9.

Congestion control A segment may be discarded during its transfer across an internet either because transmission errors are detected in the packet containing the segment or because a router or gateway along the path being


LWE
Incremented as bytes are acknowledged
UWE
Incremented by number in window field
Send window, W_S
M1
M2
M3
Bytes already acknowledged
Bytes waiting to be acknowledged
Bytes waiting to be sent in send buffer

LWE = lower window edge, initialized to (ISN + 1)
UWE = upper window edge, initialized to (ISN + 1) + advertized window
W_S = (UWE – LWE), flow stopped when $W_S = 0$
M1, 2, 3 = sending AP messages
(1), (2) etc. = segments sent by TCP entity
Note: that TCP chooses the size of segments it sends

Figure 12.9 TCP sliding window.

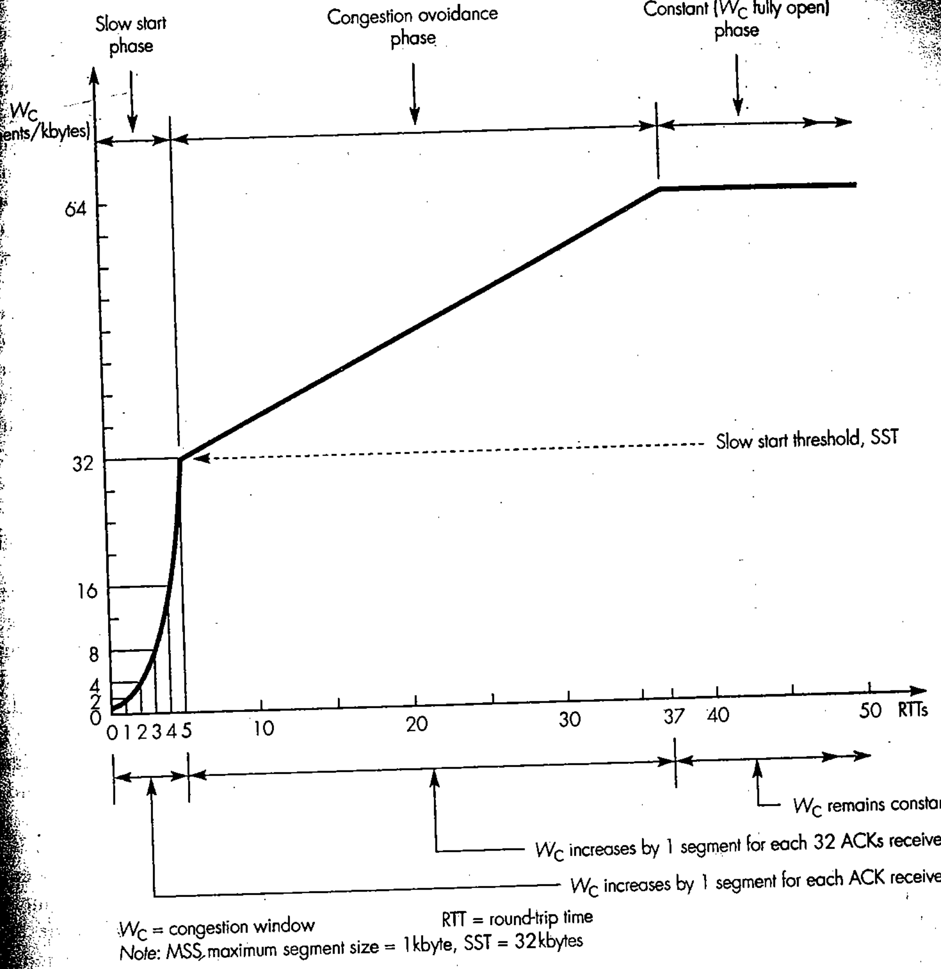
followed becomes congested; that is, during periods of heavy traffic it te rary runs out of buffer storage for packets in the output queue asso with a line. However, the extensive use of optical fiber in the transmissio work means that the number of lost packets due to transmission err relatively small. Hence the main reason for lost packets is congestion wi the internet.

To understand the reason for congestion, it should be remembered the path followed through an internet may involve a variety of different tra mission lines some of which are faster – have a higher bit rate – than others general, therefore, the overall speed of transmission of segments over the pa being followed is determined by the bit rate of the slowest line. Also, cong tion can arise at the sending side of this line as the segments relating multiple concurrent connections arrive at a faster rate than the line can tra mit them. Clearly, if this situation continues for even a relatively short ti interval, the number of packets in the affected router output queue builds until the queue becomes full and packets have to be dropped. This also affe the ACKs within the lost segments and, as we have just seen, this can have a sig nificant effect on the overall time that is taken to transmit a message.

In order to reduce the likelihood of lost packets occurring, the TCP each host has a congestion control/avoidance procedure which, for eac connection, uses the rate of arrival of the ACKs relating to a connection regulate the rate of entry of data segments – and hence IP packets – into th internet. This is in addition to the window flow control procedure which, a we have just seen, is concerned with controlling the rate of transmission o segments to the current capacity of the receive buffer in the destination hos Hence in addition to a send window variable, W_S, associated with the flo control procedure, each TCP also has a **congestion window** variable, W_C, ass ciated with the congestion control/avoidance procedure. Both ar maintained for each connection and the transmission of a segment relatin to a connection can only take place if both windows are in the open state.

As we can see from the above, under lightly-loaded network condition the flow of segments is controlled primarily by W_S and, under heavily-loade conditions, it is controlled primarily by W_C. However, when the flow of se ments relating to a connection first starts, because no ACKs have bee received, the sending TCP does not know the current loading of the interne So to stop it from sending a large block of segments – up to the agree window size – the initial size of the congestion window, W_C, is set to a singl segment which, because W_C has a dimension of bytes, is equal to the agreed MSS for the connection.

As we show in Figure 12.10, the sending TCP starts the data transfer phase of a connection by sending a single segment of up to the MSS. It then starts the retransmission timer for the segment and waits for the ACK to be received. If the timer expires, the segment is simply retransmitted. If the ACK is received before the timer expires, W_C is increased to two segments, each equal to the MSS. The sending TCP is then able to send two segments and,



igure 12.10 TCP congestion control window procedure.

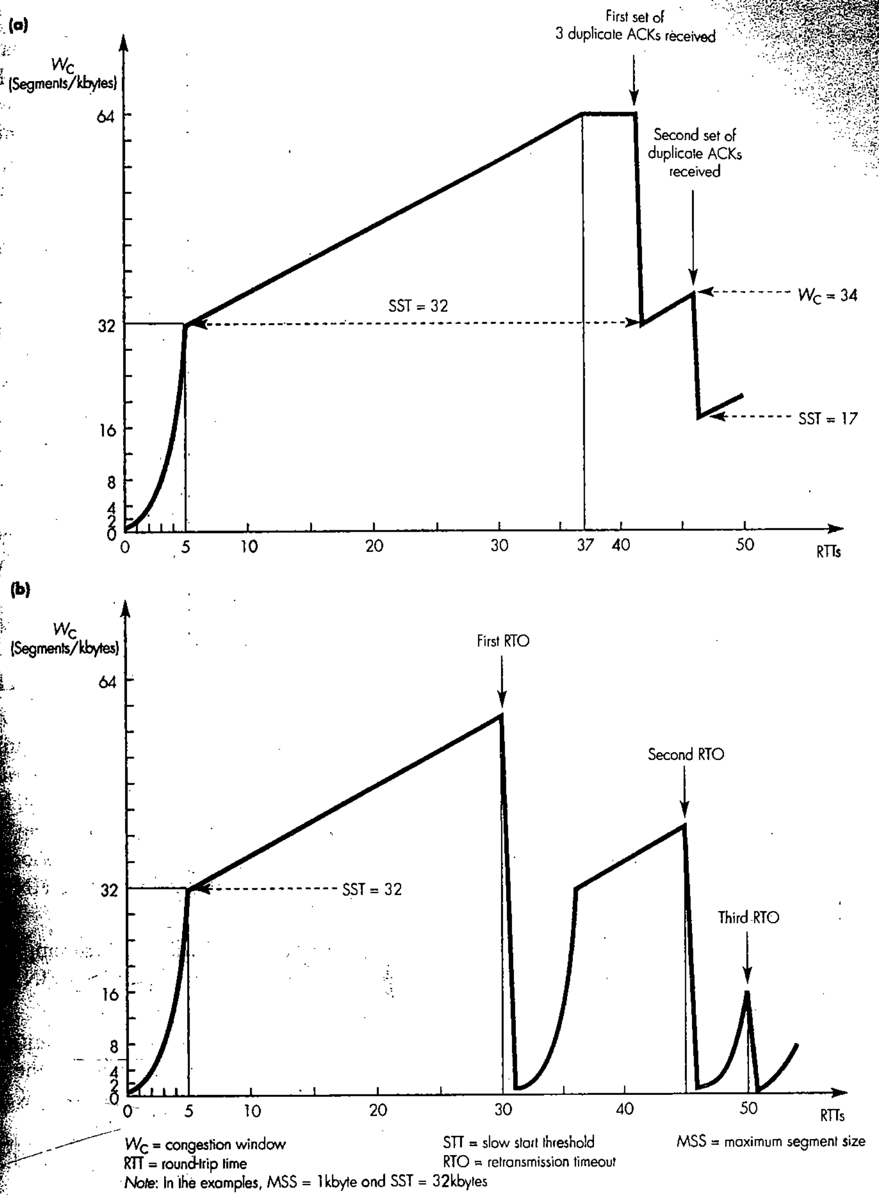
for each of the ACKs it receives for these segments, W_C is increased by one segment (MSS). Hence, the sending TCP can now send four segments and, as we can see, W_C increases exponentially. Even though W_C increases rapidly, this phase is called **slow start** since it builds up from a single segment. It continues until a timeout for a lost segment occurs, or a duplicate ACK is received, or an upper threshold is reached. This is called **the slow start threshold (SST)** and, for each new connection, it is set to 64 kbytes. In the example, however, it is assumed to be initialized to 32 kbytes which, becau the MSS is 1 kbyte, is equal to 32 segments. Assuming the SST is reached, th is taken as an indication that the path is not congested. Hence the conne tion enters a second phase during which, instead of W_C increasing by segment (MSS) for each ACK it receives, it increases by $1/W_C$ segments each ACK received. Hence, as we can see, W_C now increases by 1 segment f each set of W_C ACKs that are received. This is called the **congestion avoidan** phase and, during this phase, the increase in W_C is additive. It continues unti a second threshold is reached and, in the example, this is set at 64 kbytes. On reaching this, W_C remains constant at this value.

The profile shown in Figure 12.10 is typical of a lightly-loaded internet i which none of the lines making up the path through the internet is con gested. Providing W_C remains greater than the maximum flow contro window, the flow of segments relating to the connection is controlled prima ily by W_S. During these conditions all segments are transferred with relatively constant transfer delay and delay variation. As the number of co nections using the internet increases, however, so the traffic level increase up to the point at which packet (and hence segment) losses start to occu and, when this happens, the TCP controlling each connection starts to adju its congestion window in a way that reflects the level of congestion.

The steps taken depend on whether a lost packet is followed by duplica ACKs being received or the retransmission timer for the segment expiring. I the case of the former, as we saw earlier in Figure 12.7, the receipt of dupl cate ACKs is indicative that segments are still being received by th destination host. Hence the level of congestion is assumed to be light and, o receipt of the third duplicate ACK relating to the missing segment – fa retransmit – the current W_C value is halved and the congestion avoidan procedure is invoked starting at this value. This is called **fast recovery** and a example is shown in Figure 12.11(a).

In this example it is assumed that the first packet loss occurs when W_C at its maximum value of 64 segments which, with an MSS of 1 Kbyte, is equ to 64 kbyte. Hence on receipt of the third duplicate ACK, the lost segmen retransmitted and W_C is immediately reset to 32 segments/kbytes. The W_C then incremented back up using the congestion avoidance procedu However, when it reaches 34 segments, a second segment is lost. It is assume that this also is detected by the receipt of duplicate ACKs and hence W_C reset to 17 segments before the congestion avoidance procedure is restarted

In the case of a lost segment being detected by the retransmission tim expiring, it is assumed that the congestion has reached a level at which n packets/segments relating to the connection are now getting through. As w show in the example in Figure 12.11(b), when a retransmission timeou (RTO) occurs, irrespective of the current W_C, it is immediately reset to 1 seg ment and the slow start procedure is restarted. Thus, when the level o congestion reaches the point at which RTOs start to occur, the flow of seg ments is controlled primarily by W_C.



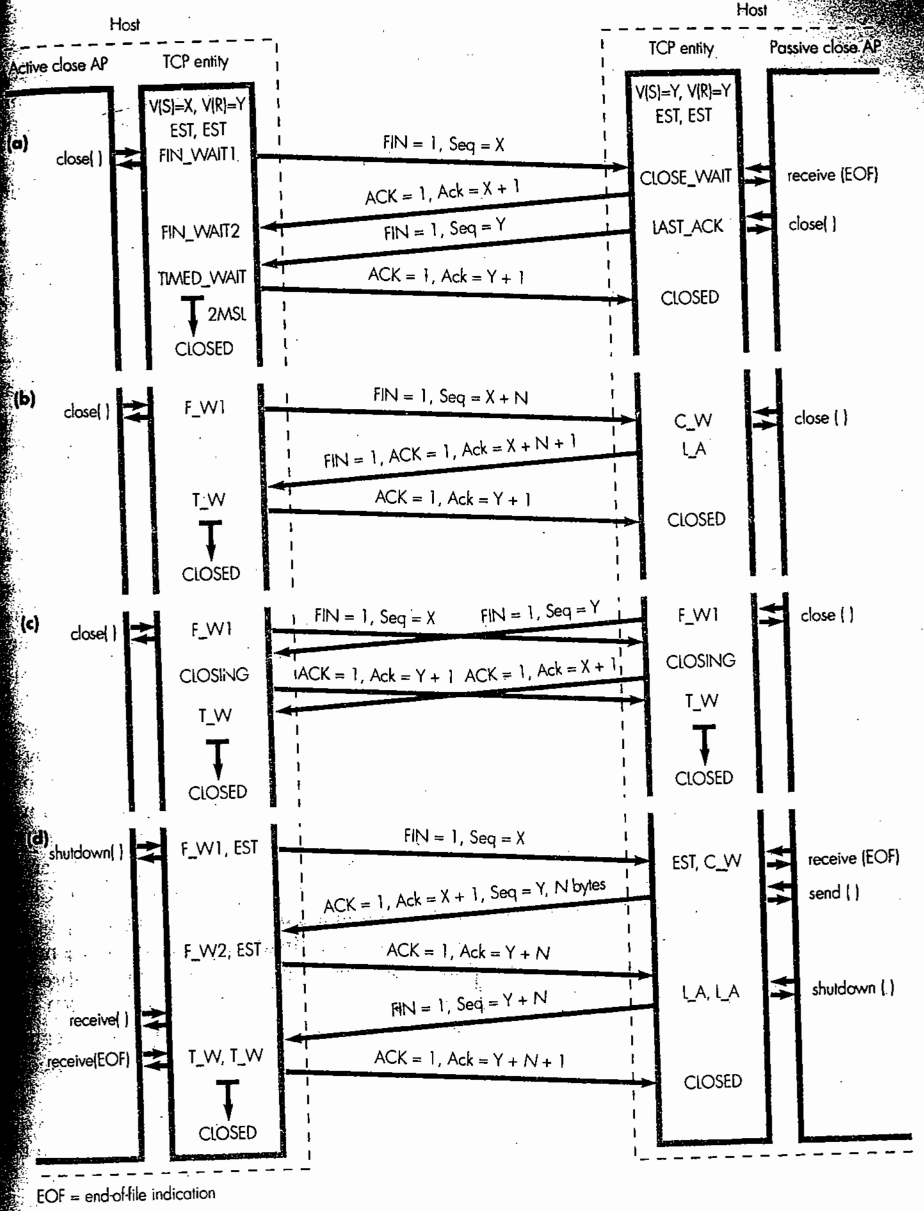
gure 12.11 TCP congestion window adjustments: (a) on receipt of duplicate ACKs; (b) on iry of a retransmission timer.

Connection termination

As we indicated earlier, each TCP connection is duplex and hence da be transferred in both directions simultaneously. To support this, each entity maintains separate send and receive sequence and window vari and a separate state variable for each direction of flow. When a conne is terminated, each direction of flow is closed separately. In practice, are a number of ways this is carried out and four examples are give Figure 12.12.

In most cases, the TCP entity at each end of a connection goes throug different sequence of states. In order to discriminate between the two en the AP which issues the first *close( )* – and hence the TCP which sends the fi FIN segment – performs what is called an **active close** procedure and t other side, a **passive close**. As we can see in the first example – part (a) – ea procedure involves a slightly different sequence:

- In all the examples, both the forward and return paths of the TCP connection are currently in the ESTABLISHED (EST) – data transfer – sta
- The connection termination is started by one of the APs – normally th client – issuing a *close( )* primitive. The TCP entity at this side then goe through the active close procedure and the TCP at the other side the passive close.
- On receipt of the *close( )* primitive, the active TCP entity sends a FIN segment – that is, a segment with the FIN bit on – with a sequence number equal to the current V(S), X. It then enters the FIN_WAIT1 st to indicate it is waiting for an ACK for the first FIN.
- On receipt of the FIN, the passive TCP first returns an ACK indicating correct receipt of the FIN and, when its local AP does the next *receive(* an end-of-file (EOF) is returned to indicate that no more data will be coming from the other side for this connection. It then enters the CLOSE_WAIT state to indicate it is waiting for a *close( )* from its local A Note that the ack number in the ACK is equal to X + 1 since a FIN consumes a byte of the byte stream.
- On receipt of the ACK, the TCP performing the active close simply enters the FIN_WAIT2 state to indicate it is now waiting for a FIN fro the passive side.
- At some time later the AP in the passive side issues a *close( )* primitive a as a result, the local TCP sends a FIN to indicate the closure of the connection in the reverse direction. It then enters the LAST_ACK stat to indicate it is now waiting for the last ACK.
- On receipt of the FIN at the active side, the TCP first returns an ACK fo the segment. It then starts a timer called the **2MSL timer** and enters the TIMED_WAIT state. When configuring each TCP entity a parameter called the **maximum segment lifetime (MSL)** is entered. This is the


Host
Active close AP
TCP entity
Host
TCP entity
Passive close AP
(a)
close( )
V(S)=X, V(R)=Y
EST, EST
FIN_WAIT1
FIN = 1, Seq = X
V(S)=Y, V(R)=Y
EST, EST
CLOSE_WAIT
receive (EOF)
ACK = 1, Ack = X + 1
FIN_WAIT2
FIN = 1, Seq = Y
LAST_ACK
close( )
TIMED_WAIT
ACK = 1, Ack = Y + 1
CLOSED
2MSL
CLOSED
(b)
close( )
F_W1
FIN = 1, Seq = X + N
C_W
L_A
close ( )
FIN = 1, ACK = 1, Ack = X + N + 1
T_W
ACK = 1, Ack = Y + 1
CLOSED
CLOSED
(c)
close( )
F_W1
FIN = 1, Seq = X
FIN = 1, Seq = Y
F_W1
close ( )
CLOSING
ACK = 1, Ack = Y + 1
ACK = 1, Ack = X + 1
CLOSING
T_W
T_W
CLOSED
CLOSED
(d)
shutdown( )
F_W1, EST
FIN = 1, Seq = X
EST, C_W
receive (EOF)
send ( )
ACK = 1, Ack = X + 1, Seq = Y, N bytes
F_W2, EST
ACK = 1, Ack = Y + N
L_A, L_A
shutdown ( )
receive( )
FIN = 1, Seq = Y + N
receive(EOF)
T_W, T_W
ACK = 1, Ack = Y + N + 1
CLOSED
CLOSED
EOF = end-of-file indication

igure 12.12 Connection close examples: (a) normal (4-way) close; (b) 3-way close; (c) simultaneous close; (d) half-close.

maximum time duration a segment can exist in the internet before b discarded. In practice, therefore, it is related to the time-to-live value used in each IP packet header. Typical values for the MSL are between seconds and 2 minutes. This means that should the ACK for the secon FIN be lost, the active TCP is still able to receive the retransmitted FIN

- Finally, on receipt of the ACK, the TCP at the passive side deletes the connection record relating to this connection and, when the 2MSL tim expires, the TCP at the active side does the same.

The segment sequence shown in Figure 12.12(b) is similar to that in first example except that at the passive side data is still waiting to be ackno edged when the *close( )* is received. Hence the passive TCP piggybacks ACK for the data in the same segment that carries the FIN in the reve direction. This occurs when the *close( )* is received before the FIN arrives an as we can see, this reduces the standard closure to a 3-way segment exchan rather than a 4-way exchange. Note that with this, however, data may be l at the passive side if both sides of the connection are closed on receipt of th *close( )* primitive.

The segment sequence shown in Figure 12.12(c) illustrates what happen when the AP in both hosts issue a *close( )* simultaneously. As we can see, in th case the TCP at both sides carries out the active close sequence. Here, how ever, the intermediate state CLOSING is entered on receipt of the FIN from the opposite side. Then, as before, on receipt of the related ACK segmen both sides enter the TIMED_WAIT state to wait for the 2MSL timer to expi before closing down.

When an AP initiates the termination of a connection with a *close( )* prim tive, this indicates to its local TCP that it has now completed sending data/messages relating to the session and expects the remote AP to do same. As a result, both TCPs proceed to close the (duplex) connection us one of the segment exchanges shown in the first three examples. With som applications, however, although the AP in the active side has completed sendi data, it still expects to receive further data from the correspondent AP. Clea since both directions of a TCP connection are managed separately, this is pos ble. Hence to enable its local TCP to discriminate between this and a norm close, the AP issues a *shutdown( )* call. The local TCP initiates the closure of side of the connection but leaves the other side in the ESTABLISHED sta This is known as a **half-close** and an example showing a typical segme sequence is shown in Figure 12.12(d). The following points should be noted

- In the figure, we show the state of both the forward and return paths of the connection at both sides and, as before, initially these are both in the ESTABLISHED (EST) state.
- On receipt of the *shutdown( )* call, the local TCP leaves the return path in the EST state but proceeds to close the forward path. Hence it sends a FIN segment and enters the FIN_WAIT1 state.

- On receipt of this, an EOF is returned to the correspondent AP in response to the next *receive( )* to indicate that it will not receive any further data. It then changes the state of this side of the connection to the CLOSE_WAIT state. The AP then receives some data to send in the return direction and, since this path is still open, it sends the data and, in the same segment, piggybacks the acknowledgment of the FIN segment.
- On receipt of this, the other TCP returns an ACK for the data and, in response to the ACK for the FIN it sent, changes the state of this side of the connection to the FIN_WAIT2 state. Also, at some time later the AP reads the data using a *receive( )* call.
- After the ACK for the data is received, a *shutdown( )* is received from the AP. This results in a FIN segment being sent for the return path and the state of both paths being set into the LAST_ACK state.
- On receipt of the FIN, an EOF is returned to the local AP in response to the next *receive( )* to inform it that no further data will be coming. An ACK for the FIN is then returned and the state of both sides of the connection changed to the TIMED_WAIT state.
- On receipt of the ACK, the TCP deletes the connection record relating to this connection and, when the 2MSL timer expires, the TCP at the other side does the same.

The sequences shown in all four examples relate to what is called an **orderly release** since both FINs are sent either with or after all outstanding data relating to the session has been sent. In some instances, however, a connection is closed abruptly by the TCP at one side sending a segment with the RST bit on. This results in both sides of the connection being closed immediately and hence any data currently being held by either TCP will be lost. This is called an **abortive release** and an example of its use is when the sequence numbers relating to the connection become unsynchronized.

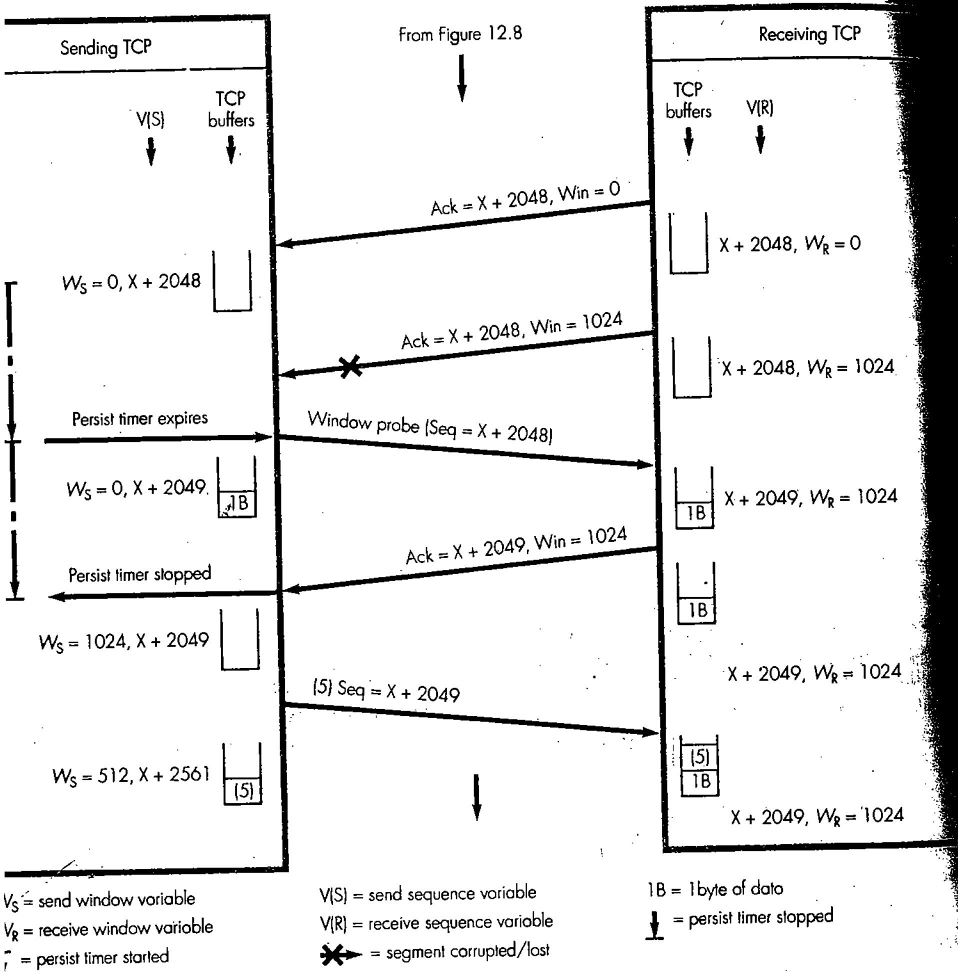
12.3.3 Additional features

The various examples we used in the data transfer section were chosen to explain the main procedures relating to this part of the TCP protocol. In addition, however, there are a number of details relating to these procedures that the examples did not show. In this section we identify a number of these.

Persist timer

In the example shown in Figure 12.8, it was assumed that the duplicate ACK containing the window update – Ack = X + 2048, Win = 1024 – was received by the sending TCP error free. In practice, of course, the packet containing this segment may have been corrupted or lost. If this had occurred, since the TCP entity at the receiving side has initiated the reopening of the window, it is now waiting to receive further segments. However, because the ACK is not received, the send window of the sending TCP is still zero and so it co to assume that it cannot send any more segments. Although the dat segments is acknowledged, acknowledgments are not. Hence if the update was not received, deadlock would occur with each side waiting other. Figure 12.13 shows how this can be avoided.

As we can see, the example relates directly to the sequence we s earlier in Figure 12.8. Whenever the sending TCP sets its send window


Sending TCP
From Figure 12.8
Receiving TCP
V(S)
TCP buffers
TCP buffers
V(R)
Ack = X + 2048, Win = 0
X + 2048, W_R = 0
W_S = 0, X + 2048
Ack = X + 2048, Win = 1024
X + 2048, W_R = 1024
Persist timer expires
Window probe (Seq = X + 2048)
W_S = 0, X + 2049
1B
1B
X + 2049, W_R = 1024
Ack = X + 2049, Win = 1024
1B
Persist timer stopped
W_S = 1024, X + 2049
X + 2049, W_R = 1024
(5) Seq = X + 2049
W_S = 512, X + 2561
(5)
(5)
1B
X + 2049, W_R = 1024
V_S = send window variable
V_R = receive window variable
= persist timer started
V(S) = send sequence variable
V(R) = receive sequence variable
= segment corrupted/lost
1B = 1byte of data
= persist timer stopped

igure 12.13 Persist timer: application and operation.

zero, it starts a timer. This is known as the **persist timer**, and if a segment containing a window update is not received before the timer expires, the sending TCP sends what is called a **window probe** segment. Even though the send window is zero, a TCP can always send a single byte of data. Hence the probe segment contains the first byte of the remaining data and, as we can see, if the ACK for this has a nonzero window value, then the flow of data segments can be resumed. Alternatively, should the value still be zero, then the timer is restarted and the procedure repeats. This is repeated at 60 s intervals until either a nonzero window value is received or the connection is terminated.

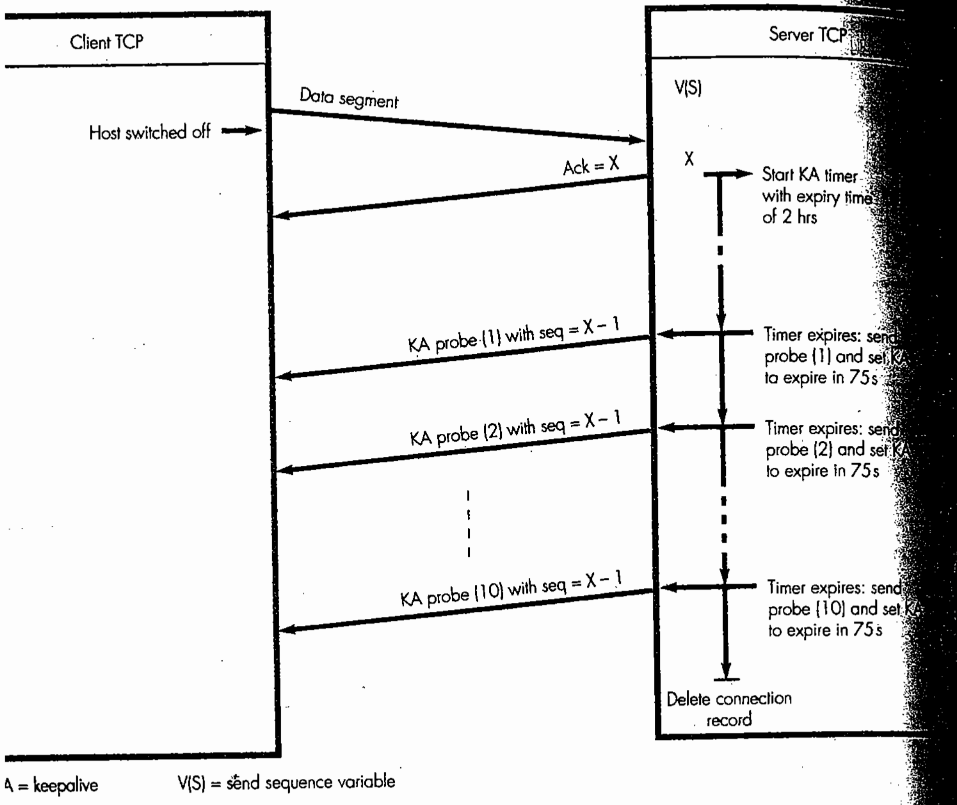
Keepalive timer

Once a connection between two TCP entities has been set up, it remains in place until the connection is terminated by the two communicating APs. As we saw in the previous section, this involves the two APs initiating the closure of their side of the connection. In most client–server applications, however, if the client host is simply switched off (instead of going through the normal log off procedure) then the connection from the server to the client will remain in place even though the client host is no longer responding. To overcome this, although not part of the TCP specification, many TCP implementations in servers include a timer known as the **keepalive timer**. The way this is used is shown in the example in Figure 12.14.

A separate timer is kept by the server TCP for all the connections – of which there can be many – that are currently in place. The default value of the keepalive timer is two hours and, should no data segments be exchanged over a connection during this time interval, the TCP in the server sends a probe segment to the client and sets the timer this time to 75 s. The probe segment has no data and has a *sequence number* of one less than the current V(S) of the server–client side of the connection. If the client host is still switched on – and the TCP connection is still in place – the client TCP simply returns an ACK for this with its current V(R) within it and, on receipt of this, the server TCP restarts the keepalive timer to 2 hours. If the client TCP does not respond, then the timer expires and the TCP in the server sends a second probe with the timer again set to 75 s. This procedure is repeated and, if no reply is received after 10 consecutive probes, the client is assumed to be switched off (or unreachable) and the server terminates the connection.

Silly window syndrome

The combined IP and TCP packet/segment headers are at least 40 bytes. Hence the larger the number of bytes in the data field of each segment, the smaller are the overheads associated with each transfer and the higher is the mean end-to-end data transfer rate. In the flow control example shown earlier in Figure 12.8, it was assumed that the AP at the receiving side read large (1024-byte) blocks of data from the receive buffer which, in turn, enabled the sending TCP to operate efficiently by always sending segments containing large amounts of data. In some interactive applications, however, a condition


Client TCP
Server TCP
Data segment
Host switched off
V(S)
Ack = X
X
Start KA timer with expiry time of 2 hrs
KA probe (1) with seq = X − 1
Timer expires: send probe (1) and set KA to expire in 75s
KA probe (2) with seq = X − 1
Timer expires: send probe (2) and set KA to expire in 75s
KA probe (10) with seq = X − 1
Timer expires: send probe (10) and set KA to expire in 75s
Delete connection record
A = keepalive
V(S) = send sequence variable

gure 12.14 Keepalive timer: application and operation.

can arise which results in a very small number of bytes being sent in each
ment. It is known as the **silly window syndrome (SWS)** and can arise eith
the receiving side or the sending side.

To avoid this occurring, an added feature defined by Clark is inco
rated into the basic flow control procedure we discussed earlier. At
receiving side this forbids a receiving TCP entity from advertising a sm
window and, at the sending side, it forbids a sending TCP entity from send
segments containing small amounts of data. Note that both features are co
plementary to the Nagle algorithm we discussed earlier and, in ma
instances, both operate together.

To illustrate how the problem can arise at the receiver, assume that in a
interactive application a server AP initiates the transfer of a large characte
file to a client AP but the latter proceeds to read it (from the receive buffer o

the receiving TCP) a single character at a time. On receipt of the first block of characters/bytes sent by the TCP in the server – up to the current maximum window size – the receiving TCP in the client would first return an ACK for the complete block but with a window value of zero. Then, assuming the procedure we showed in the earlier example in Figure 12.8, each time the client AP reads a character from the TCP receive buffer, the client TCP will return a window update to inform the TCP at the server side that it can now send a further character/byte. This it duly does and, on its receipt, the client TCP returns an ACK for it but again with a window value of zero. Similarly, when the next character is read by the AP, the client TCP returns a second window update of 1 byte/character and the procedure repeats until the complete file has been transferred in this way.

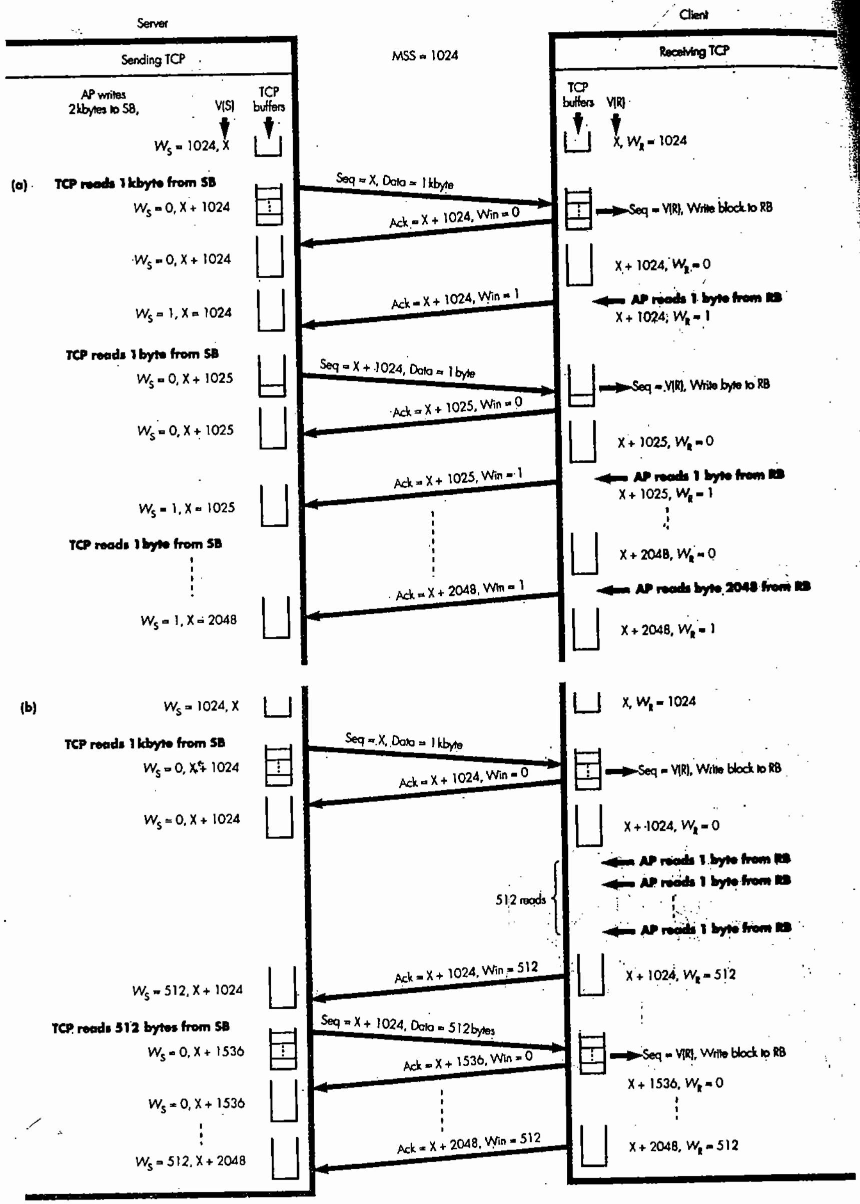
To avoid this happening, a receiving TCP is prevented from sending a window update until there is sufficient space in its buffer either for a segment equal to the maximum segment size in use or, one half of the maximum buffer capacity – and hence window capacity – whichever is the smaller. An example showing this is given in Figure 12.15. Part (a) illustrates the problem and part (b) shows how the solution attributed to Clark avoids the problem occurring.

In the example it is assumed that the window size in use is 1024 bytes and the MSS is also 1024 bytes. Hence, as we can deduce from part (a), after the first 1024-byte block has been transferred, without the Clark extension, each of the remaining 1024 bytes is transferred in a separate segment. Moreover, each of these may require an ACK and a window update segment. With the Clark extension, however, as we can see from part (b), after the first 1024 bytes have been transferred the remaining bytes are transferred in just two 512-byte segments; the choice of 512 being half of the buffer capacity.

As we indicated earlier, the same problem can arise at the sending side. For example, in an interactive application in which a user enters a string of characters at a keyboard, the sending AP may write each character into the send buffer as it is entered. In the absence of the Clark extension, the sending TCP may then proceed to send each character in a single segment. With the Clark extension, however, the sending TCP is made to wait until it has accumulated a sufficient number of bytes/characters to fill a segment equal to either the MSS in use with the connection or, one half of the buffer capacity of the receiving TCP. The latter is determined from the maximum window update value the sending TCP has received from the receiving TCP.

Window scale option

During our discussion of the HDLC protocol in Section 6.8, we saw that for transmission links which have a large bandwidth/delay product, the sizes of the send and receive sequence number fields in the frame header are extended in order to allow a larger window size to be used. As we saw in Example 6.8, this improves the utilization of the available link bandwidth. This can also be a requirement with TCP when the transmission path


Server
Sending TCP
MSS = 1024
Client
Receiving TCP
AP writes 2kbytes to SB,
V(S)
TCP buffers
TCP buffers
V(R)
W_S = 1024, X
X, W_R = 1024
(a)
TCP reads 1kbyte from SB
Seq = X, Data = 1kbyte
Seq = V(R), Write block to RB
W_S = 0, X + 1024
Ack = X + 1024, Win = 0
W_S = 0, X + 1024
X + 1024, W_R = 0
Ack = X + 1024, Win = 1
AP reads 1 byte from RB
X + 1024; W_R = 1
W_S = 1, X = 1024
TCP reads 1byte from SB
Seq = X + 1024, Data = 1byte
W_S = 0, X + 1025
Seq = V(R), Write byte to RB
Ack = X + 1025, Win = 0
W_S = 0, X + 1025
X + 1025, W_R = 0
Ack = X + 1025, Win = 1
AP reads 1 byte from RB
X + 1025, W_R = 1
W_S = 1, X = 1025
TCP reads 1byte from SB
X + 2048, W_R = 0
Ack = X + 2048, Win = 1
AP reads byte 2048 from RB
W_S = 1, X = 2048
X + 2048, W_R = 1
(b)
W_S = 1024, X
X, W_R = 1024
TCP reads 1kbyte from SB
Seq = X, Data = 1kbyte
W_S = 0, X + 1024
Seq = V(R), Write block to RB
Ack = X + 1024, Win = 0
W_S = 0, X + 1024
X + 1024, W_R = 0
AP reads 1 byte from RB
AP reads 1 byte from RB
512 reads
AP reads 1 byte from RB
Ack = X + 1024, Win = 512
W_S = 512, X + 1024
X + 1024, W_R = 512
TCP reads 512 bytes from SB
Seq = X + 1024, Data = 512bytes
W_S = 0, X + 1536
Seq = V(R), Write block to RB
Ack = X + 1536, Win = 0
X + 1536, W_R = 0
W_S = 0, X + 1536
Ack = X + 2048, Win = 512
W_S = 512, X + 2048
X + 2048, W_R = 512

ure 12.15 Silly window syndrome example: (a) the problem; (b) Clark's solution.

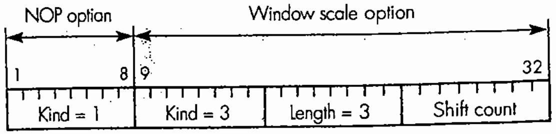
followed through the Internet covers a large distance – and hence has a large signal propagation delay – and a high mean bit rate.

As we saw in Figure 12.4, the *window size* field in the segment header is 16 bits which allows a maximum window size of 65 535 bytes. However, when the path involves intercontinental lines, for example, a typical propagation delay is 40 ms. Also, since optical fiber is now widely used, mean bit rates in excess of 155 Mbps are common. This means that, in order to fully utilize the available transmission capacity, a window size in excess of 775 000 bytes is required. Hence even though a single connection may only use a portion of the available bit rate, it can be seen that, for connections which span large distances, a larger maximum window size is required.

In order to achieve this without changing the format of the segment header, an option has been defined that enables a scaling factor to be applied to the value specified in the *window size* field. This is called the **window scale option** and is defined in **RFC 1323**. The format of this option is shown in Figure 12.16.

Normally, the option is included in the SYN segment when a connection is being established and, since the actual window size is determined by the size of the receive buffer, a different scaling factor can be agreed for use in each direction. Also, since the option is only 3 bytes in length, it is always preceded by a single byte containing a value of 1. This is known as a **no operation (NOP) option** and they are used as pad bytes so that all options are multiples of 4 bytes.

The scaling factor is defined in the *shift count* field of the option. As we can see, this is a 1-byte field and the count value can be between 0 – no scaling in use – and 14. Although the *window size* field in each segment header is only 16 bits, the send and receive window variables kept by each TCP entity are both 32-bit values. The actual window value is then computed by first writing the 16-bit value from the *window size* field into the corresponding window variable and then shifting this left – and hence multiplying it by 2 – by the number in the *shift count* field. A shift count of 1, therefore, means that the


NOP option
Window scale option
1
8
9
32
Kind = 1
Kind = 3
Length = 3
Shift count

NOP = no operation option: used to pad the window scale option to 4 bytes. The value in the *shift count* field is the power of 2 multiples of the value in the *window size* field. The maximum count value is 14.

Shift count = 0 , no scaling: maximum window = 65535 (bytes)
Shift count = 1 , multiply by 2^1: maximum window = 131 070
Shift count = 14, multiply by 2^{14}: maximum window = 1073725440

Figure 12.16 Window scale option format.

maximum window size can be up to 65 535 × 2^1 = 131 070 bytes and a shift count of 14, a maximum size of 65 535 × 2^{14} = 1 073 725 440 bytes. As we indicated earlier, however, the maximum window size that can be used in each direction is determined by the size of the corresponding receive buffer. Normally, therefore, the shift count to be used for each direction is chosen by the TCP entity since it knows the amount of memory that has been allocated for the receive buffer.

Time-stamp option

During our discussion of the error control procedure, it was assumed that the measurement procedure used to estimate the worst-case round-trip time (RTT) – used by a TCP to compute the retransmission timeout (RTO) interval – is carried out for every data segment that is sent. Although this is the case in some implementations, in others a measurement update occurs only for one segment per window. In general, this is sufficient for transfers involving a small window size – and hence small number of segments per window – but, with a large window, the use of a single segment per window can lead to a poor estimate of the RTT. As we indicated, this can result in many unnecessary retransmissions. Hence to obtain a more accurate estimate of the RTT, the **time-stamp option** is used with these implementations when a large window size is detected. The option is defined in **RFC 1323** and allows the sending TCP to obtain an estimate of the RTT with each ACK it receives.

The option is requested by the TCP that performs an active open, by including the time-stamp option in the SYN segment. The request is then accepted if the receiving TCP includes a time-stamp option in the SYN segment that it returns. Once accepted, both TCPs can then include a time-stamp option in every data segment that they send. The format of the option is shown in Figure 12.17(a) and, as we can see, since it is 10 bytes long, the option is preceded by two NOP option bytes.

The TCP at each side of a connection keeps a 32-bit *timeout timer* which it uses to estimate the RTT for its own direction of transmission. The timer at each side is independent – not synchronized – and is incremented at intervals of between 1 ms and 1 s, a typical interval being 500 ms. For every data segment it sends, the sending TCP reads the current time from the timer and writes this into the *time-stamp value* field. Then, when the TCP returns an ACK for a segment, it writes the same time-stamp value in the *time-stamp echo reply* field. Hence on receipt of each ACK segment, the sending TCP can estimate the RTT for its direction of the connection by computing the difference between the current time in its timer and the time-stamp value contained in the option field of the ACK.

This procedure will work providing the receiving TCP returns an ACK for each data segment it receives. In many implementations, however, as we showed earlier in Figure 12.8, an ACK may be returned only after multiple data segments have been received. Indeed, following the widespread introduction of optical fiber, most TCP implementations now return an ACK for

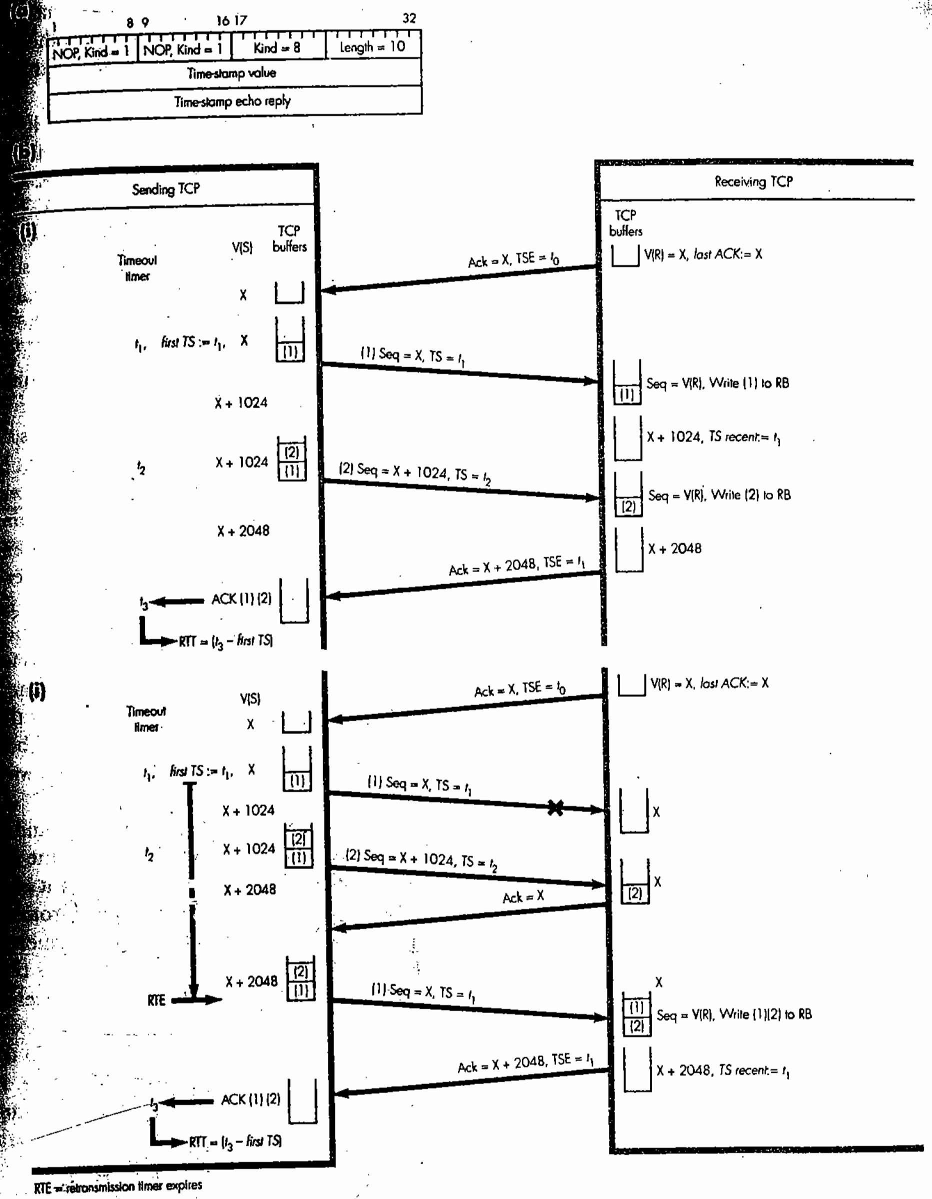

(a)
1 8 9 16 17 32
NOP, Kind = 1
NOP, Kind = 1
Kind = 8
Length = 10
Time-stamp value
Time-stamp echo reply
(b)
Sending TCP
Receiving TCP
(i)
Timeout timer
V(S)
TCP buffers
TCP buffers
Ack = X, TSE = t_0
V(R) = X, last ACK:= X
X
t_1, first TS := t_1, X
(1) Seq = X, TS = t_1
Seq = V(R), Write (1) to RB
X + 1024
X + 1024, TS recent:= t_1
t_2
(2) Seq = X + 1024, TS = t_2
Seq = V(R), Write (2) to RB
X + 2048
Ack = X + 2048, TSE = t_1
t_3 ACK (1) (2)
RTT = (t_3 – first TS)
(ii)
Ack = X, TSE = t_0
V(R) = X, last ACK:= X
(1) Seq = X, TS = t_1
(2) Seq = X + 1024, TS = t_2
Ack = X
RTE
(1) Seq = X, TS = t_1
Seq = V(R), Write (1)(2) to RB
Ack = X + 2048, TSE = t_1
X + 2048, TS recent:= t_1
RTE = retransmission timer expires

Figure 12.17 Time-stamp option: (a) option format; (b) two examples.

every other data segment received. In such cases, the question a which of the time-stamps – one from each of the data segments th received – does the receiving TCP return in the *echo reply* field of The answer is the time-stamp from the first in-sequence segment receiving TCP received after it returned the last ACK.

To implement this, the receiving TCP maintains two variables in nection record: *lastACK* and *TSrecent*. Each time the receiving TCP re ACK, it keeps a record of the V(R) which it sent in the ACK in *lastAC* when the first data segment arrives after it has returned the ACK sequence number in the segment header equals that stored in *las* keeps a record of the time-stamp value it contains in *TSrecent*. As each quent data segment arrives, the TCP simply processes it in the nor and, when it returns an ACK, includes the current value in *TSrecent* in th *reply* field of the option. Thus, the RTT computed by the sending T reflect that the receiving TCP is only returning an ACK on receipt of m data segments.

If the sequence number in a segment is not that expected – that sequence number is greater than V(R) – this indicates that a segme been lost. In this case, the receiving TCP simply returns an ACK and pro to wait for the missing segment to be received. Then when it arrives, the stamp from this is echoed, not the time-stamp from the out-of-seq segment. To allow for the possibility of the first segment in a new seq being corrupted or lost, the sending TCP keeps a record of the time sent the segment in a variable called *firstTS*. Should the segment retransmitting, the value in *firstTS* is used as the *time-stamp value*. In th the corrupted RTT also includes the time to retransmit the seg Although this is an overestimate of the RTT, it is considered to be bette an underestimate, as would have been the case if the later time had used. Two examples of sequences that show the two alternatives are sho Figure 12.17(b).

In the first example, it is assumed that no lost segments occur ACK is returned on receipt of every other data segment. The time segment is sent, t_1, is stored in *firstTS* and, at the receiving side, since segment expected, *TSrecent* becomes equal to the time-stamp value fro segment, t_1. Then, when the ACK is returned after the second seg received, the value in the *time-stamp echo* field is set to t_1. The RTT computed as the difference between the time when the ACK was re and the value in *firstTS*, t_1.

In the second example, it is assumed that the first data segment new sequence is corrupted/lost but the second is received error fre this case, *TSrecent* is not updated until the second copy of the retransmi first segment is received. However, since this has the same *time-stamp val* the corrupted first segment, the *time-stamp echo* is the same as befo Hence, as we can see, the computed RTT includes the time taken to retra mit the first segment.

SACK-permitted option

As we indicated earlier, for connections that involve paths through the Internet which span large distances, the propagation delay – also referred to as latency – of the path can be several tens of milliseconds. Hence in addition to influencing the choice of window size, it also has an impact on the efficiency of the error control scheme. As we showed in the example in Figure 12.7, to allow for segments being received out of sequence, a segment is retransmitted only after three duplicate ACKs have been received. Clearly, with connections that have a large RTT associated with them, the delays involved each time a packet/segment is lost or corrupted can be large. In order to reduce this delay, the alternative selective repeat/acknowledgment error control scheme can be used. This is defined in **RFC 2018** and is requested by including the **SACK-permitted option** in the SYN segment header. It is then accepted if the receiving TCP includes the same option in the SYN segment that it returns.

We explained the principle of operation of the selective repeat error control scheme in Section 6.7.3 and gave two example frame sequences showing how the protocol overcomes both a corrupted I-frame and a corrupted ACK-frame. A similar protocol is used with TCP SACK except in this, the SACK segment returned by a receiving TCP contains a list of the data segments that are missing in a specified window of data. In this way, all the missing segments are retransmitted in a single RTT.

Protection against wrapped sequence numbers

When very large amounts of data are being transferred between two hosts using a high bit rate LAN, for example, to speed up the transfer, the window scale option is often used with the maximum shift count of 14. An example is performing a backup of the contents of a disk over a LAN where, since this may involve many gigabytes, it is possible for the sequence numbers to wrap around during the backup. This means that a segment that is lost during one pass through the sequence numbers may be retransmitted and received during a later pass through the numbers so ruining the integrity of the transfer.

To overcome this possibility, the **PAWS** – protection against wrapped sequence numbers – algorithm is often used. This involves the time-stamp option being selected and both sides using the current 32-bit time-stamp as an extension to the 32-bit sequence number. Effectively, this produces 64-bit sequence numbers which overcomes the problem.

12.3.4 Protocol specification

To finish our discussion of TCP we shall illustrate how the segment sequences shown in the various examples, coupled with the socket primitives shown in Table 12.1, relate directly to the formal specification of the TCP protocol entity. Recall from Section 6.7.7 that protocol specifications are carried out in a number of ways. One of the most widely used methods is to use a combination of a state transition diagram and an extended event–state table. A state transition diagram illustrates the various sequences in a pictorial form a hence the various examples relate directly to this. The extended event–sta table method, however, is better for implementing a protocol since, as we s in the example in Section 6.7.7, program code can be derived directly fro this. To avoid too much detail, we shall consider only the specification of th connection establishment and termination phases of a basic client TCP an the related server TCP since, as we saw in Figure 6.33, the data transfer pha contains many state variables and predicates.

Using the methodology we established in Section 6.7.7, the various incoming events, protocol states, outgoing events, and specific (internal) actions fo both the client and server are shown in parts (a) and (b) of Table 12.2 respectively. As we can see, these relate directly to the segment sequence shown i Figure 12.5(a) – connection establishment – and Figure 12.12(a) – connection termination. The specification of both the client TCP and the server TCP protocols in both state transition diagram and extended event–state diagram form are then shown in parts (a) and (b) of Figure 12.18 respectively.

Table 12.2 Abbreviated names used in the specification of the TCP protocol: (a) client TCP; (b) server TCP.

(a) Incoming events

Name	Interface	Meaning
connect ( )	AP_user	Initiate the setting up of a connection
SYN+ACK	IP_provider	SYN+ACK received from server (TCP)
DATA	IP_provider	Block of data received from server
close( )	AP_user	Initiate the closure of a connection
ACK	IP_provider	ACK received from server
2MSL	Timer	2MSL timer expires
FIN	IP_provider	FIN received from server
RST	IP_provider	Abort connection

States

Name	Meaning
CLOSED	No connection in place
SYN_SENT	SYN sent to server (TCP)
ESTABLISHED	SYN received from server, connection now in place
FIN_WAIT1	FIN sent to server
FIN_WAIT2	ACK to FIN received from server
TIMED_WAIT	FIN received from server

Table 12.2 continued

Outgoing events

Name	Interface	Meaning
SendSYN	IP provider	Send SYN to server (TCP)
SendACK	IP provider	Send ACK to server
SendDATA	IP provider	Send block of data to server
SendFIN	IP provider	Send FIN to server

Specific actions

[1] Block client AP
[2] Unblock client AP
[3] Write EOF to receive buffer (RB)
[4] Start 2MSL timer

(b) Incoming events

Name	Interface	Meaning
listen ( )	AP_user	Create queue for connection requests
accept ( )	AP_user	Block server AP
SYN	IP_provider	SYN received from client
RST	IP_provider	RST received from client
ACK	IP_provider	ACK received from client
DATA	IP_provider	DATA received from client
FIN	IP_provider	FIN received from client
RST	IP_provider	Abort connection

States

Name	Meaning
CLOSED	No connection in place
LISTEN	Server AP blocked, TCP waiting for a SYN
SYN_RCVD	SYN received from a client AP
ESTABLISHED	ACK for own SYN received, connection now in place
CLOSE_WAIT	FIN received from client TCP
LAST_ACK	FIN sent and waiting for an ACK

▶

Table 12.2 Continued

Outgoing events		
Name	**Interface**	**Meaning**
SendSYN+ACK	IP_provide	[illegible]
SendDATA	IP_provide	[illegible]
SendFIN	IP_provide	Send FIN to clien
SendACK	IP_provide	Send ACK to clien

Specific actions	
[1] Create queue for connection request	[3] Unblock serve
[2] Block server AP	[4] [illegible]

It should be stressed that the specifications are only simplified versions the real specifications. For instance, they do not include the actions taken the event of simultaneous connection requests or simultaneous closes occu ring. Also, when in the ESTABLISHED state, many predicates – for examp to check whether sequence numbers are valid and within the current windo – and state variables are used when determining the required action and ne state. Again, a good insight into the latter can be obtained from the exampl sequences associated with the data transfer phase.

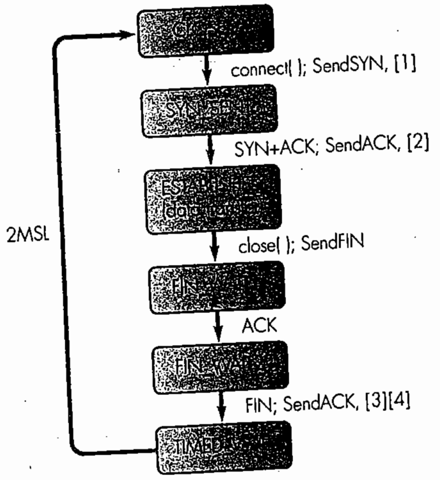
12.4 UDP

Recall that with TCP there is no correlation between the size of the mes sages/blocks of data submitted by a user AP and the amount of data in eac TCP segment that is used to transfer the messages. Typically, as we saw i Section 12.2, the latter is determined by the path MTU to avoid fragment tion of each segment occurring.

In contrast, with UDP each message/block of data that is submitted by a user AP is transferred directly in a single IP datagram. On receipt of the message, the source UDP simply adds a short header to it to form what is called a **UDP datagram**. This is then submitted to the IP layer for transfer over the internet using, if necessary, fragmentation. At the destination, the IP first determines from the *protocol* field in the datagram header that the destination protocol is UDP, and then passes the contents of the (IP) datagram to the UDP.

The latter first determines the intended user AP from a field in the UDP datagram header and then passes the contents of the (UDP) datagram to the peer user AP for processing. There are no error or flow control procedures involved and hence no connection set up is required. The service offered by UDP to a user AP, therefore, is simply an extension of the service provided by IP. Hence in addition to two-party calls, multicast group calls can be supported. Nevertheless, the set of service primitives and the protocol are both simpler than those of TCP. As with TCP, we shall discuss the services and the protocol separately.

(a)



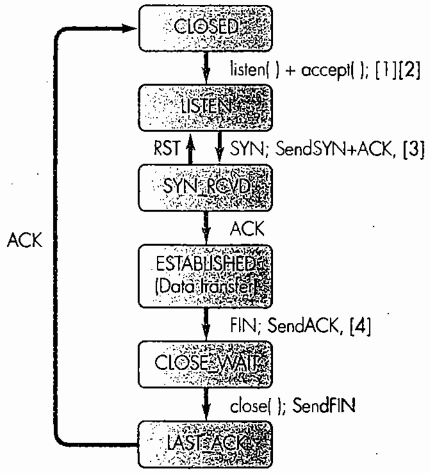
Present state \ Incoming event	connect( )	SYN+ACK	close( )	ACK	FIN	2MSL
CLOSED	1	0	0	0	0	0
SYN_SENT	0	2	0	0	0	0
ESTABLISHED	0	0	3	0	0	0
FIN_WAIT1	0	0	0	4	0	0
FIN_WAIT2	0	0	0	0	5	0
TIMED_WAIT	0	0	0	0	0	6

0 = SendRST, CLOSED
2 = SendACK, [2], ESTABLISHED
4 = FIN_WAIT2
6 = CLOSED

1 = SendSYN, [1], SYN_SENT
3 = SendFIN, FIN_WAIT1
5 = SendACK, [3][4], TIMED_WAIT

igure 12.18 TCP protocol specifications: (a) client; (b) server.

(b)



Present state \ Incoming event	listen( ) + accept( )	SYN	RST	ACK	FIN	close( )
CLOSED	1	0	0	0	0	0
LISTEN	0	2	0	0	0	0
SYN_RCVD	0	0	3	4	0	0
ESTABLISHED	0	0	0	0	5	0
CLOSE_WAIT	0	0	0	0	0	6
LAST_ACK	0	0	0	7	0	0

0 = SendRST, CLOSED
2 = SendSYN+ACK, [3], SYN_RCVD
4 = ESTABLISHED
6 = SendFIN, LAST_ACK
1 = [1][2], LISTEN
3 = LISTEN
5 = SendACK, [4], CLOSE_WAIT
7 = CLOSED

12.18 Continued.

12.4.1 User services

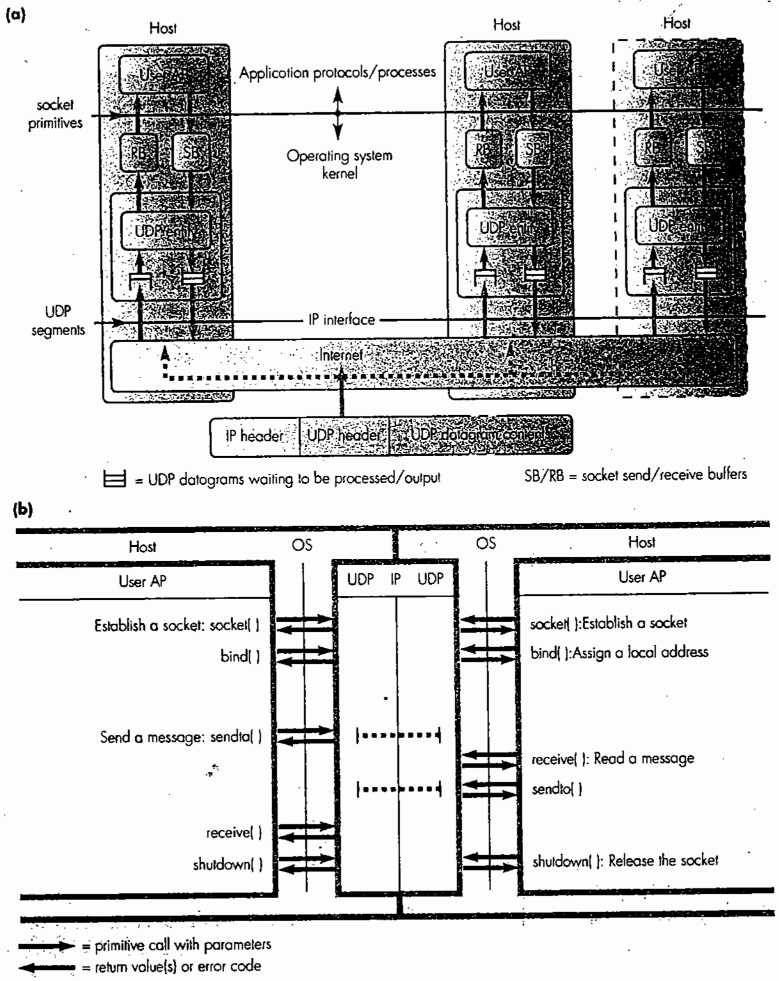
As with TCP, the most widely used set of user service primitives associated with UDP are the Berkeley Unix socket primitives. With most applications that use UDP, the two user APs either exchange messages on a request-response basis or simply initiate the transfer of blocks of data as these are generated. A typical list of service primitives – system/function calls – is given in Table 12.3 and their use is shown in diagrammatic form in Figure 12.19.

Table 12.3 List of socket primitives associated with UDP and their parameters

Primitive	*Parameters*
socket( )	service type, protocol, address format; return value = socket descriptor or error code
bind( )	socket descriptor, socket address (= host IP address + port number) return value = success or error code
sendto( )	socket descriptor, local port number, destination port number, destination IP address, precedence, pointer to message buffer containing the data to send, data length (in bytes); return value = success or error code
receive( )	socket descriptor, pointer to message buffer into which the data should be put, length of the buffer, return value = success/error code or end-of-file (EOF)
shutdown( )	socket descriptor, return value = success or error code

As we can see in Figure 12.19(b), prior to exchanging any messages, each of the user APs involved in the call must first establish a socket between itself and its local UDP. The parameters associated with the *socket( )* primitive include the service required (datagram service), the protocol (UDP), and the address format (Internet). Once a socket has been created – and send/receive memory buffers allocated – a socket descriptor is returned to the AP which it then uses with each of the subsequent primitive calls. If this is the only AP that is running in the host, the AP can now start to send and receive messages. If not – for example a server is involved – the AP then issues a *bind( )* primitive which, in addition to the socket descriptor, has an address parameter. This comprises the IP address of the host plus the 16-bit port number the AP wishes to be assigned to the socket. In the case of a server AP, for example, this will be the related well-known port number. When a *bind( )* is not used, the port number will be an ephemeral port number and assigned locally.

Once a socket has been created, the user AP can start to send and receive messages. However, since no connection is involved – and hence no connection record has been created – in addition to the message, the AP must specify the IP address of the destination host – or the IP multicast address in the case of multiple destinations – and the port number of the destination socket/AP. Also, if required, it must specify a precedence value to be sent in the *type of service* field of the IP datagram header. Hence, as we can see in Table 12.3, the *sendto( )* primitive includes each of these fields in its set of parameters. Finally, when all the data transfers associated with the call/session have been carried out, the socket is released by issuing a *shutdown( )* call.


(a)
Host
Host
Host
socket primitives
Application protocols/processes
Operating system kernel
RB
SB
UDP entity
UDP segments
IP interface
Internet
IP header
UDP header
UDP datagram content
= UDP datagrams waiting to be processed/output
SB/RB = socket send/receive buffers
(b)
Host
OS
OS
Host
User AP
UDP
IP
UDP
User AP
Establish a socket: socket( )
bind( )
Send a message: sendto( )
receive( )
shutdown( )
socket( ):Establish a socket
bind( ):Assign a local address
receive( ): Read a message
sendto( )
shutdown( ): Release the socket
= primitive call with parameters
= return value(s) or error code

2.19 UDP socket primitives: (a) socket interface; (b) primitives and their use.

12.4.2 Protocol operation

The format of each UDP datagram is shown in Figure 12.20(a). The *source port* is the port number of the sending application protocol/socket and the *destination port* is that of the peer (receiving) application protocol(s). Both are 16-bit integers. The value in the *length* field is the number of bytes in the complete (UDP) datagram and includes the 8-byte header and the contents of the *data* field. The *checksum* covers the complete datagram, header plus contents. In addition, as with TCP, since only a simple checksum is used to compute the checksum value in the IP header, in order to add an additional level of checking, some selected fields from the IP header are also included in the computation of the UDP checksum. The fields used form what is called the **UDP pseudo header**. These are identified in Figure 12.20(b) and, as we can see, they are the source and destination IP addresses and the protocol value (=17 for UDP) from the IP header, plus the value from the length field in the UDP header.

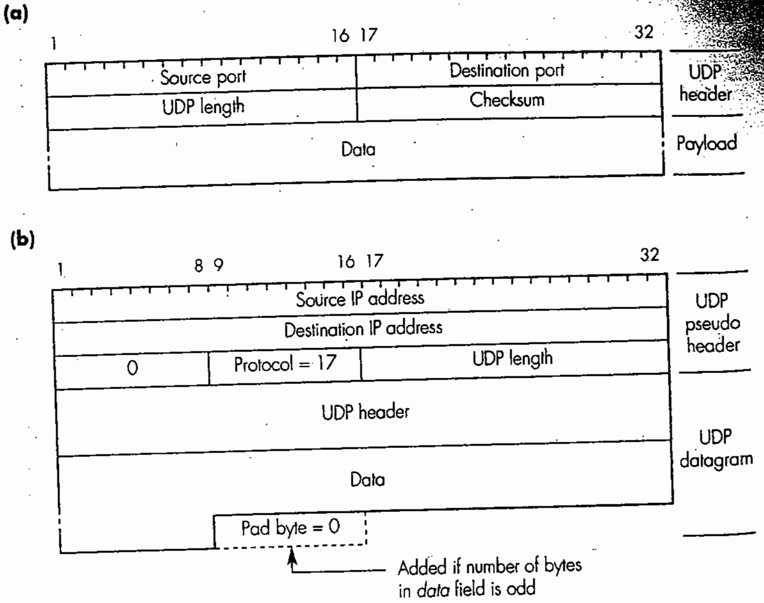

(a)
1
16 17
32
Source port
Destination port
UDP length
Checksum
UDP header
Data
Payload
(b)
1
8 9
16 17
32
Source IP address
Destination IP address
0
Protocol = 17
UDP length
UDP pseudo header
UDP header
Data
UDP datagram
Pad byte = 0
Added if number of bytes in data field is odd

Figure 12.20 UDP datagram format: (a) UDP header fields; (b) fields used in pseudo header for computation of checksum.

The computation of the UDP checksum uses the same algorithm as that used by IP. As we saw in Section 9.2, this is computed by treating the complete datagram as being made up of a string of 16-bit words which are added together using 1s complement arithmetic. Using 1s complement number representation, a value of zero is represented as either all 1s or all 0s. With UDP, however, if the computed 1s complement sum is all 0s, then the checksum is sent as all 1s. This is done because the use of a checksum with UDP is optional and, if the value in the checksum field is all 0s, this indicates the sender has not sent a checksum.

Since the number of bytes in the original UDP data field – and he submitted application protocol data unit – may be odd, in order to ens the same checksum is computed by both UDPs, a pad byte of zero is adde the data field whenever the number of bytes in the original data field is o Thus the value in the length field must always be an even integer. Also, sin the UDP datagram is carried in a single IP datagram, as we saw in Section 9 in order to avoid fragmentation, the size of each submitted application pro col data unit must be limited to that dictated by the MTU of the pa followed through the Internet by the IP datagram. For example, assuming path MTU of 1500 bytes, allowing for the 8 bytes in the UDP header and 2 bytes in the IP header, the maximum submitted application PDU should b limited to 1472 bytes if fragmentation is to be avoided.

Finally, although the maximum theoretical size of a UDP datagram – a determined by the maximum size of an IP datagram which is 65 535 (64K – 1 bytes – is 65 507 (65 535 – 20 – 8) bytes, the maximum value supported b most implementations is 8192 bytes or less.

12.5 RTP and RTCP

As we showed in Figure 12.1 and explained in the accompanying text, whe an application involves the transfer of a real-time stream of audio and/o video over a packet network – for example the speech relating to an Interne phone call – the timing information that is required by the receiver to outp the received packet stream at the required rate is provided by the real-tim transport protocol (RTP). In addition, for applications that involve bo audio and video streams – for example the audio and video associated with videophone call – the real-time transport control protocol (RTCP) is used synchronize the two media streams prior to carrying out the decoding opera tion. We showed a schematic diagram illustrating the use of both thes protocols in Figure 5.6 and in this section we describe their main features.

12.5.1 RTP

As we showed in Figure 5.6, normally, the audio and/or video associated wit an application are digitized separately using a particular codec. In addition when the bitstreams are to be transported over a packet network like an inte net, each of the bitstreams must be sent in the form of a stream of packets using, for example, the UDP protocol. Similarly, at the receiver, the bitstream must be reconstructed from the stream of received packets. During their transfer over an internet, however, some packets may be lost and/or delayed by varying amounts. Also, since the packets may follow different paths through the internet, they may arrive at the destination in a different order. Hence before the reconstructed bitstream can be passed to the decoder, any missing packets must be detected and compensated for. Similarly, any delay variations in the packet arrival times must be allowed for. These are the functions performed by RTP and a schematic diagram illustrating its use is shown in Figure 12.21(a).

The packet format used with RTP is shown in Figure 12.21(b). The *version* (*V*) field indicates the version of RTP that is being used, *P* is a pad bit and *X* an extension flag to allow extensions to the basic header to be defined and added in the future.

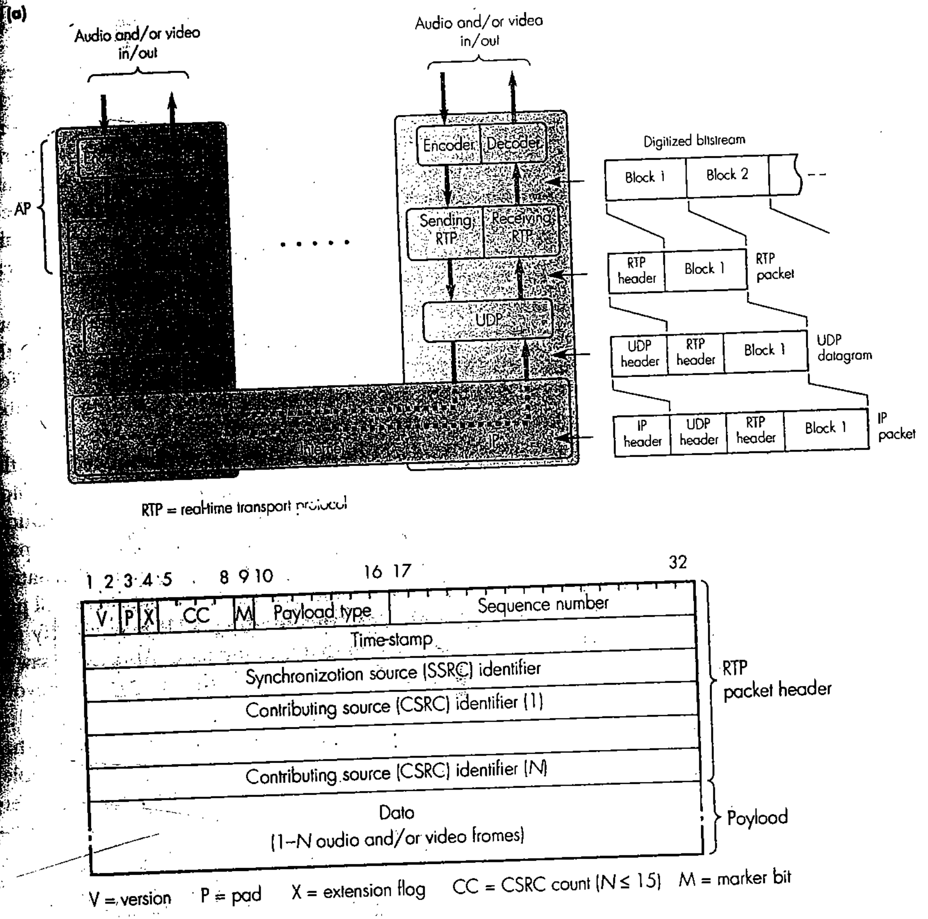

(a)
Audio and/or video in/out
AP
Audio and/or video in/out
Encoder
Decoder
Sending RTP
Receiving RTP
UDP
IP
Digitized bitstream
Block 1
Block 2
RTP header
Block 1
RTP packet
UDP header
RTP header
Block 1
UDP datagram
IP header
UDP header
RTP header
Block 1
IP packet
RTP = real-time transport protocol
1 2 3 4 5
8 9 10
16 17
32
V
P
X
CC
M
Payload type
Sequence number
Time-stamp
Synchronization source (SSRC) identifier
Contributing source (CSRC) identifier (1)
Contributing source (CSRC) identifier (N)
Data
(1–N audio and/or video frames)
RTP packet header
Payload
V = version
P = pad
X = extension flag
CC = CSRC count (N ≤ 15)
M = marker bit

Figure 12.21 Real-time transport protocol: (a) usage; (b) packet format.

In a multicast call/session, each of the participants that con the session – rather than passively listening – is called a **contributing** (**CSRC**) and is uniquely identified by means of a 32-bit identifier wh cally, is the IP address of the source. During a multicast session the stream from multiple sources may be multiplexed together for trans purposes by a device known as a **mixer**. Hence the resulting RTP packe contain blocks/frames of digitized information from multiple sources enable the receiver to relate each block/frame to the appropriate p pant, the CSRC identifier for each block/frame is included in the hea the new packet. The number of CSRC identifiers present in the pack given in the *CSRC count* (*CC*) field. Since this is a 4-bit field, up to 15 tributing sources – and hence CSRC identifiers – can be present in the packet header.

As we saw in Chapter 4, the bitstream produced by the different type audio and video codecs is made up of a sequence of blocks or frames with a unique start and end delimiter. Associated with the *marker*(*M*) bi profile which enables the receiver to interpret the packet data on the co block/frame boundaries. Also, since there is a range of different audio video codecs, the *payload type* field indicates the type of encoder that has be used to encode the data in the packet. Moreover, since each packet conta this field, the type of encoder being used can be changed during a c should the QoS of the network being used change.

Each packet contains a *sequence number* which is used to detect lost or o of-sequence packets. In the case of a lost packet being detected, normally, t contents of the last correctly received packet are used in its place. The effe of out-of-sequence packets being received is overcome by buffering a numb of packets before playout of the data they contain starts.

The value in the *time-stamp* field indicates the time reference when th packet was created. It is used to determine the current mean transmission delay time and the level of jitter that is being experienced. This information together with the number of lost packets, forms the current QoS of the pat through the network/internet. As we shall see, periodically this information is returned to the sending RTP by the related real-time transport control pro tocol. Then, should the QoS change, the sending RTP may modify th resolution of the compression algorithm that is being used. Also, as we saw Figure 1.22, the level of jitter is used to determine the size of the playou buffer that is required.

The *synchronization source* (*SSRC*) *identifier* identifies the source device that has produced the packet contents. In a videoconferencing call, for example the data generated by each contributing source may be from multiple different devices – a number of microphones, cameras, computers, and so on – and the SSRC indicates from which device the source information has come. The receiving RTP then uses the SSRC to relay the reconstructed bitstream to the related output device interface.

As we can deduce from Figure 12.21(a), on receipt of each IP packet, the various fields in the UDP datagram header are used within the destination host to deliver the RTP packet to the receiving RTP entity. The various fields in the RTP packet header then enable the receiving RTP to reconstruct the bitstream for each device decoder associated with the session.

12.5.2 RTCP

As we can conclude from the previous section, RTP is concerned with the transfer of the individual streams of digitized data associated with a multimedia call/session. The real-time transport control protocol (RTCP) then adds additional system-level functionality to its related RTP such as the means for a receiving RTP to integrate and synchronize the individual packet streams together and for a sending RTP to be informed of the currently-prevailing network QoS. Hence as we showed in Figure 5.6, the RTCP operates alongside of RTP and shares information with it. However, as we saw in Figure 5.8, each RTCP has a different (UDP) port number associated with it so that it can operate independently of RTP.

The RTCP in all of the systems involved in a call/session, periodically exchange messages with one another. Each message is sent in a RTCP packet to the same network address – but with the RTCP port number – as the RTP to which the message relates. The general scheme is shown in Figure 12.22 and the messages that are exchanged relate to:

- *integrated media synchronization*: in applications that involve separate audio and video streams that need to be integrated together, as we showed in Figure 5.18, a common system time clock is used for synchronization

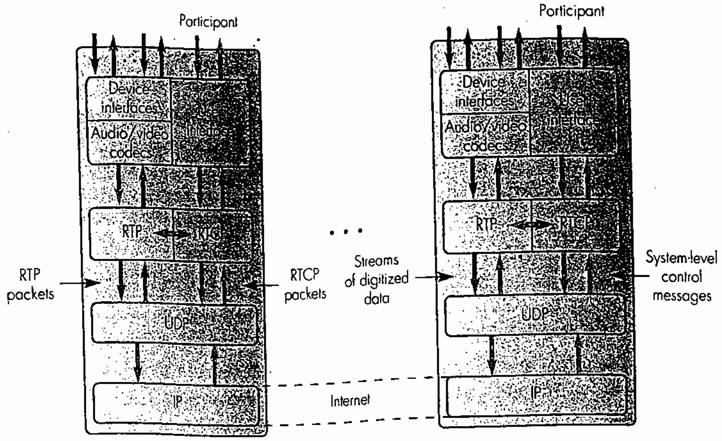


Figure 12.22 Real-time transport control protocol (RTCP) usage.

purposes. Normally, the system that initiates the call/conference provide this function – or sometimes a separate reference time server – and the RTCP in all of the other systems then exchange messages with the RTCP in this system so that they are utilizing the same system time clock;

- *QoS reports*: as we indicated earlier, the number of lost packets, the level of jitter, and the mean transmission delay are continuously computed by each RTP for the packet streams they receive from all of the other contributing sources. The adjoining RTCP then sends a message containing the related information to the RTCP in each of these systems at periodic intervals. The RTCP in each of these systems then performs any system-level functions that may need to be performed; for example changing the resolution of the compression algorithm or the size of the playout buffer;
- *participation reports*: these are used during a conference call, for example to enable a participant to indicate to the other participants that it is leaving the call. Again, this is done by means of the RTCP in the participant's system. The participant enters an appropriate message and this is then sent via RTCP to the RTCP in each of the other systems. Typically, a related message is then output on the screen of each of the other systems;
- *participation details*: information such as the name, email address, phone number, and so on of each participant is sent to all of the other participants in a RTCP message. In this way, each of the participants knows the identity and contact information of all of the other participants.

Summary

In this chapter we have described four protocols that are used widely with packet networks to support applications. Both TCP and UDP are transport layer protocols while RTP and RTCP, despite their name, both form an integral part of an application layer protocol/process. A summary of the main features of each protocol is given in Figure 12.23.

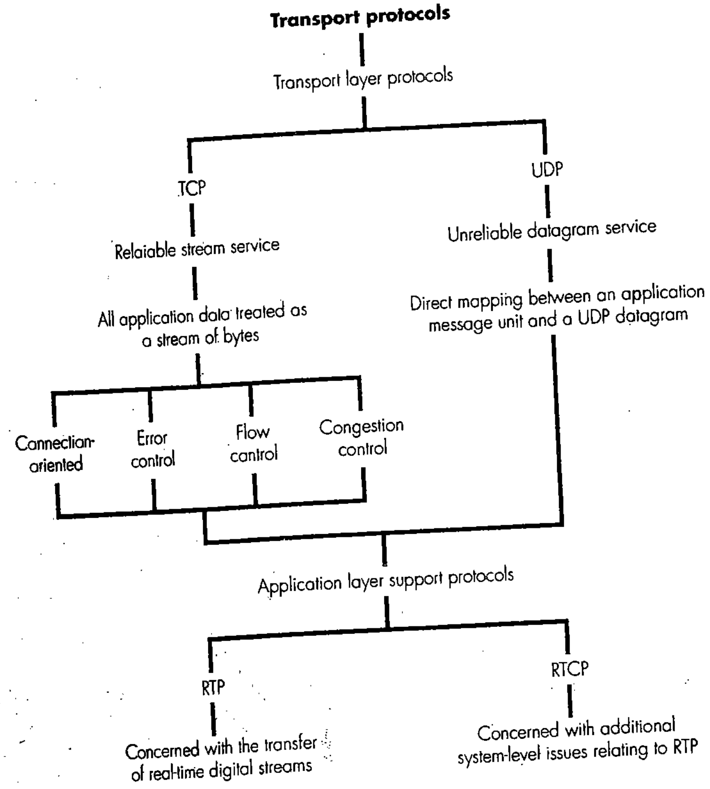

Transport protocols
Transport layer protocols
TCP
Relaiable stream service
All application data treated as a stream of bytes
Connection-oriented
Error control
Flow control
Congestion control
UDP
Unreliable datagram service
Direct mapping between an application message unit and a UDP datagram
Application layer support protocols
RTP
Concerned with the transfer of real-time digital streams
RTCP
Concerned with additional system-level issues relating to RTP

Figure 12.23 Transport protocols chapter summary.

Exercises

Section 12.2

12.1 By means of a diagram, show the position of the TCP, UDP, RTP, and RTCP, protocols in relation to the various network layer and application protocols in the TCP/IP protocol suite. Include in your diagram a received message that contains the headers relating to the various protocols and indicate the interlayer address selectors that are used to route the application data in the received message to the intended application protocol/process.

12.2 In relation to the port numbers that are present in the TCP and UDP protocol headers, explain the meaning of the terms "ephemeral" and "well-known port numbers".

12.3 With the aid of a diagram showing how two application protocols/processes communicate with each other using the TCP/IP protocol suite, explain the meaning of the terms:
- (i) end-to-end communication,
- (ii) TCP/UDP service primitives,
- (iii) network-interface protocol,
- (iv) hop-by-hop transfer.

Section 12.3

12.4 Explain the meaning of the term "reliable stream service" in relation to the operation of TCP and why a logical connection between the two communicating TCP entities must be established to provide this service.

12.5 In relation to the data transfer phase of TCP, explain the meaning of the terms:
- (i) segment,
- (ii) maximum segment size (MSS),
- (iii) path MTU.

12.6 State why both flow control and congestion control procedures are required with TCP.

Section 12.3.1

12.7 With the aid of a diagram, explain the meaning of the following terms relating to TCP:
- (i) socket interface,
- (ii) socket primitives,
- (iii) application program interface (API),
- (iv) TCP (protocol) entity.

12.8 List the set of four socket primitives that ar issued by a server AP to carry out a passiv open; that is, to establish a socket betwee itself and its local TCP entity. Explain the us of each primitive and the meaning of the term "socket descriptor" and "socket address".

12.9 By means of a diagram, show the set of socke primitives that are issued by a client AP t carry out an active open and those that ar issued by the server in response to these Describe the effect of each primitive at both the client and server sides and, in relation to the *connect( )* primitive, the parameters that are associated with it.

12.10 Explain the meaning and use of the following relating to a socket:
- (i) socket descriptor,
- (ii) socket address,
- (iii) pass-through parameter.

12.11 State why the TCP entity in both a client and a server needs to create and maintain a connec- tion record for each new (TCP) connection. List the main fields that are present in a con- nection record and describe their use.

12.12 Associated with a socket is a send buffer and a receive buffer. Explain the use of each and how they relate to the segments that are sent/received by the two TCP entities. Include in your explanation how an AP can force the transmission of a small block of data.

Section 12.3.2

12.13 Use the TCP segment format given in Figure 12.4(a) to explain the use of the following:
- (i) source and destination port numbers,
- (ii) sequence and acknowledgment numbers,
- (iii) code bits,
- (iv) window size,
- (v) urgent pointer,
- (vi) options and the header length.

4 Describe the role of the checksum field in a TCP segment header. Include in your description why additional header fields are added to those in the TCP header and why pad bytes are used.

15 With the aid of a time sequence diagram, explain how a logical connection between two TCP entities is established using a three-way handshake procedure. Include in your diagram the socket primitives at both the client and server side that trigger the sending of each segment. Also, explain how the initial sequence number in each direction is selected.

16 Identify and explain briefly the essential procedures that are followed by two correspondent TCP entities during the data transfer phase.

17 With the aid of a time sequence diagram, show how the individual characters entered by a user at a keyboard are transferred over a TCP connection using immediate acknowledgments. Hence quantify the overheads associated with this mode of operation.

12.18 Use a time sequence diagram to show how the overheads identified in Exercise 12.17 are reduced using
- (i) delayed acknowledgments, and
- (ii) the Nagle algorithm.

Explain the reason why the Nagle algorithm is the preferred mode of working with interactive applications over the Internet.

12.19 Explain why additional buffers to the send and receive buffers associated with the socket are required within the TCP entity at each side of a connection.

12.20 Explain why the sending TCP entity only retransmits a segment if it receives three duplicate acknowledgments for the segment. Include in your explanation the role of the retransmission timer used by the sending TCP entity and the meaning of the term "fast retransmit".

12.21 The following questions relate to the time sequence diagram shown in Figure 12.7.
- (i) Assuming no segments had been corrupted/lost, deduce how many Data and ACK segments would have been sent.
- (ii) Assuming segment (2) had not been corrupted/lost, deduce the segments that would have been sent by both TCP entities.
- (iii) Explain why the second copy of segment (2) is discarded but an ACK is returned.
- (iv) Assuming the MSS for the (TCP) connection was 1024 bytes and one (data) segment was corrupted, deduce how many data and ACK segments would be sent.

12.22 State the meaning of the term "retransmission timeout (RTO) interval" in relation to to TCP connection and why it is necessary to determine this dynamically for each new connection.

12.23 Explain the operation of the exponential backoff algorithm that is used to derive the RTO for a TCP connection. Also explain the retransmission ambiguity problem that can arise with this.

12.24 Explain the early Jacobson algorithm that was used to derive the RTO for a TCP connection. Describe the refinement that was later made to it.

12.25 Explain the meaning of the term "delayed ACK timer" in relation to TCP and why this is required.

12.26 State the relationship between the value in the *window size* field and the value in the *acknowledgment* field in the header of a TCP segment. How is the value in the *window size* field determined?

The following relate to the time sequence diagram shown in Figure 12.8.
(i) Why does the Win(dow) field in segments sent by the sending TCP remain the same?
(ii) Why does the V(R) in the receiving TCP stay at X until after the first four segments have been received?
(iii) Why does the second ACK returned by the receiving TCP contain a Win = 1024 instead of Win = 2048?
(iv) Why is the second ACK called a window update?

With the aid of a diagram, explain the meaning of the following terms relating to the TCP window flow control procedure. Include in your diagram a sequence of messages being sent each of which requires multiple segments to send:
(i) lower window edge,
(ii) upper window edge,
(iii) sliding window.

What is the main reason for lost packets in the Internet? Hence explain how each host endeavors to minimize this effect. Include in your explanation the meaning of the term "congestion window".

Explain in a qualitative way the relationship between the send window, W_S, and the congestion window, W_C, that are maintained by the TCP entity in a host for each (TCP) connection. Include in your explanation why W_c is initialized to 1 segment.

In relation to the graph shown in Figure 12.10, explain the meaning of the following terms:
(i) slow start phase, including why this is shown to increase exponentially
(ii) slow start threshold and the congestion avoidance phase, including why this is shown to increase linearly
(iii) constant phase.

The graph shown in Figure 12.10 relates to a lightly-loaded internet. During these condi-
tions, what effect does the conges
window, W_C, have on the send window, W

12.33 Explain in a qualitative way why the adj
ments to the W_c for a connection depend
whether a lost packet is detected as a resul
duplicate ACKs being received or the retra
mission timer expiring.

12.34 With the aid of a graph showing the variati
of W_c as a function of the connection R
show and explain how W_c is affected by:
(i) a set of 3 duplicate ACKs being recei
for a segment
(ii) the retransmission timer for a segme
expiring.

12.35 Use the time sequence diagrams shown
Figure 12.12(a) and (d) to explain th
sequence of segments that are exchanged wi
(i) a normal connection close,
(ii) a half close.
Include in your explanations why tw
FIN_WAIT states and a final TIMED_WA
state are required.

12.36 Explain the role of the persist timer. Hen
with the aid of the time sequence diagra
shown in Figure 12.13, explain how the eff
of a corrupted ACK containing a windo
update is overcome.

12.37 Explain the role of the keepalive tim
Hence with the aid of the time sequence di
gram shown in Figure 12.14, explain how th
effect of a client host being switched of
overcome.

12.38 Explain how the phenomenon called the sil
window syndrome can arise and the steps th
are taken at the client and server sides
avoid this happening. Using the two tim
sequence diagrams shown in Figure 12.1
explain how the Clark extension prevents th
from occurring.

12.39 By means of an example, show how for tran
mission paths that have a large bandwidth
delay product a window size field of 16 bi
may be insufficient. Hence by means of a
example, describe how the window scale option used in the TCP segment header can overcome this problem.

12.40 The computation of the RTO for a connection using the method in Exercise 12.24 required the RTT to be computed for every data segment. Explain how, with a large window size, this can lead to a poor estimate of the RTT.

Hence, with the aid of the time sequence diagram shown in Figure 12.17, explain how the number of updates can be reduced by using a time-stamp option field. Include in your explanation the use of the *time-stamp value* and *time-stop echo reply* fields and how the receiving TCP overcomes the fact that not all segments are acknowledged.

12.41 Explain the principles behind:
(i) the SACK-permitted option
(ii) how protection against wrapped sequence numbers can be overcome by using the time-stamp option.

12.42 Use the time sequence diagrams associated with the connection established (Figure 12.5) and connection close (Figure 12.12) procedures to follow the state transitions that occur at the client and server sides shown in Figure 12.18(a) and (b) respectively.

Section 12.4

12.43 What are the main differences between UDP and TCP?

12.44 By means of a diagram, show the socket interface associated with UDP in relation to a user AP. Include in your diagram the send and receive buffers associated with the socket and the input and output buffers associated with the UDP entity.

12.45 Show on a diagram a typical sequence of socket primitives that are issued at both the sending and the receiving sides to:
(i) establish a socket connection,
(ii) exchange a single UDP datagram,
(iii) release the socket connection.
Identify the main parameters associated with each primitive.

12.46 In relation to the UDP datagram format shown in Figure 12.20(b), explain how the checksum is corrupted.

State why the maximum size of UDP datagram is often less than the theoretical maximum.

Section 12.5

12.47 Describe the use of the RTP protocol and, by means of a diagram, show its position in relation to the TCP/IP protocol stack.

12.48 In relation to the RTP packet format shown in Figure 12.21(b), explain the meaning and use of the following fields:
(i) CC and CSRC,
(ii) M and payload type,
(iii) sequence number,
(iv) time-stamp,
(v) SSRC.

12.49 Describe the use of the RTCP protocol and, by means of a diagram, show its position in relation to the TCP/IP protocol stack.

12.50 Identify and give a brief explanation of the four main functions performed by RTCP.